ATARI®

Home Computer
ATARI

MICROSOFT BASIC II

Manuale di Riferimento




A Warner Communications Company

Indice

PAGINA

PAGINA

Indice Analitico

E

F

G

V

W

X

Y

Z

Elenco di Consultazione Parole Chiave

0 Prefazione

Per facilitare la consultazione, è stato compilato un elenco completo delle parole chiave organizzato per ordine alfabetico con a fianco il relativo numero di pagina.

Questo manuale descrive tutti i comandi e le istruzioni usate dal Microsoft BASIC II della Atari.
Il linguaggio BASIC sviluppato al Dartmouth College, negli Stati Uniti da John Kemeny e Thomas Kurtz, è stato concepito come un linguaggio di programmazione facile da imparare e da usare. Molte estensioni fatte negli ultimi anni hanno fatto del BASIC un linguaggio completo ed utile per programmatori esperti.

Lo scopo di questo manuale è quello di fornire agli utenti dei Home Computer delle Atari, uno strumento di veloce consultazione ed una guida per l'utilizzo del linguaggio da parte di programmatori che abbiano già acquisito una certa conoscenza operativa. Le finalità di questo Manuale non sono didattiche, ed esso neppure si propone come un testo di introduzione al linguaggio Microsoft BASIC II della ATARI.

IMPORTANTE: I programmi sviluppati utilizzando la versione base su disco del Microsoft BASIC Atari, possono essere eseguiti anche con il Microsoft BASIC II.

1 La Programmazione Creativa

Il Microsoft BASIC II della Atari è il più avanzato linguaggio di programmazione in BASIC disponibile su una singola cartuccia ROM (Read-Only Memory) da 16 K per utilizzo su Home Computer della Atari. Sarà sufficiente inserire la cartuccia nell'Home Computer Atari per rendersi conto della sorprendente versatilità e velocità operativa del Microsoft BASIC II.

Una delle prime cose che si notano nel Microsoft BASIC II è la sua semplice gestione delle stringhe. Infatti, è ora possibile usare stringhe ad una dimensione senza doverle dichiarare in anticipo al calcolatore. Il Microsoft BASIC II si spinge ancora oltre, permettendo l'uso di matrici multidimensionali di variabili e stringhe. I numeri di riga del programma possono essere inseriti automaticamente con il comando AUTO e si possono cancellare uno o più numeri di riga con DELETE.

I trattini (-) delimitano l'intervallo dei numeri di riga quando si usano i comandi LIST o DELETE. Se l'utente non è soddisfatto dei numeri di riga del programma dopo una lunga sessione, può rinumerare le righe con il comando RENUM. Inoltre, il Microsoft BASIC II usa vari comandi che vanno ad interessare i files DOS:KILL, LOCK, UNLOCK e NAME. Questi sono solo alcuni dei nuovi comandi che si possono utilizzare all'interno del programma Microsoft BASIC II.

Quando si introduce una riga di programma da tastiera, non viene fatto alcun controllo di sintassi. Il Microsoft BASIC II controlla l'esistenza di eventuali errori durante l'esecuzione del programma, permettendo di rintracciare tali errori direttamente alla loro sorgente con i comandi TRON e TROFF. Naturalmente non fa piacere a nessuno essere sorpresi a fare errori, però, visto che è nella natura delle cose, il Microsoft ve li identifica in modo quasi umano, e cioè ve li propone come semplici messaggi in chiaro.

Altri vantaggi del Microsoft BASIC sono la precisione a virgola mobile (fino a 16 cifre) e la possibilità di determinare i tipi di variabili: intera, reale a semplice precisione, reale a doppia precisione(DEFINT, DEFSNG, o DEFDBL) e esadecimale. Lo standard per costanti e variabili è reale a semplice precisione; si può comunque modificare la precisione di una variabile semplicemente accodando al suo nome "%" per gli interi e "#" per i reali a doppia precisione.

Inoltre il Microsoft BASIC II esegue funzioni matematiche più rapidamente utilizzando un interprete dedicato invece delle routine del sistema operativo ROM.

Ed è appunto per questo tipo di versatilità che il Microsoft BASIC II si è guadagnato un'eccellente reputazione tra i programmatori più esperti. Il Microsoft BASIC offre all'utente le più avanzate tecniche di programmazione in BASIC. Per esempio, con il comando MOVE si possono spostare intere sezioni di memoria da una parte all'altra. I programmatori di visualizzazioni grafiche semplicemente aggiungono il comando FILL alla loro tavolozza di metodi di programmazione. E' stata ulteriormente potenziata la funzione espressa dal comando SOUND; si può ora stabilire la durata di persistenza di un suono.

Se per una qualsiasi ragione volete inserire un intervallo nell'esecuzione del corso di un programma (anche 24 ore), basta inserire il comando AFTER. Sono pure disponibili vari comandi di OPTION. Con OPTION BASE si può porre a 1 il valore standard iniziale per l'indice di una matrice, anche se normalmente lo standard di avvio è 0. Il comando OPTION PLM riserva spazio nella memoria RAM per i grafici dei "player-missiles", mentre OPTION RESERVE permette di riservare automaticamente memoria per speciali routines in linguaggio macchina; OPTION CHR può essere usato per riservare memoria per utilizzare diverse fonti di caratteri.

La lista potrebbe continuare ancora e ancora! Si possono definire funzioni speciali con DEF e rendere più versatili il disegno e la stampa con i comandi PLOT ... TO, SCRN$ E PRINT AT. Il comando PRINT USING può fornire 12 diversi formati di schermo e di stampa per gestire numeri, i punti decimali e numerosi altri prospetti o necessità di fincati o formati particolari.
Il Microsoft BASIC II della Atari apre la porta ad un nuovo mondo di programmazione: la programmazione creativa.

[] DIMENSIONAMENTO DEL SISTEMA

Per usare la cartuccia Microsoft BASIC II, è necessaria una configurazione minima comprendente un Home Computer ATARI con 16 K di RAM (Random Access Memory) e un apparecchio televisivo o un monitor. Se si vogliono caricare o salvare i programmi su cassetta o dischetto, è anche necessario un Registratore Atari oppure una Unità a Dischi Atari. Il dischetto microsoft BASIC II, in versione estesa, richiede una Unità a Dischi e può essere usato solo con il Sistema Operativo per Disco (DOS) ATARI, Versione 2.0S.

[] CARICAMENTO DEL MICROSOFT BASIC II

Se il sistema non è fornito di unità a dischi, occorre seguire le seguenti istruzioni per caricare il Microsoft BASIC II:

1. Inserire la cartuccia Microsoft BASIC II della Atari nell'apposita fessura.

Premere la cartuccia con attenzione esercitando una pressione costante.

2. Accendere l'Home Computer ATARI posizionando su ON l' interruttore dell'alimentazione che si trova sul lato destro della console.

3. A questo punto il Microsoft BASIC II viene caricato nella memoria del calcolatore.

Se il sistema è fornito di unità a dischi, il caricamento del Microsoft BASIC II con il dischetto di Estensione avviene nel seguente modo:

1. Assicurarsi che l'interruttore dell'alimentazione dell'Home Compu Atari sia disattivato.

2. Accendere l'Unità a Dischi N° 1. Attendere che l'indicatore luminoso rosso BUSY si spenga.

3. Inserire il dischetto di Estensione Microsoft BASIC II nell' unità dischi. Assicurarsi che l'etichetta sia rivolta verso l'esterno e a destra.

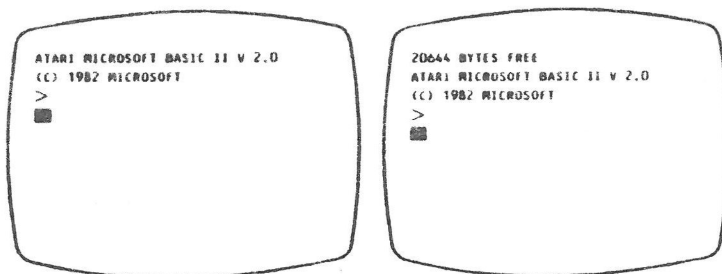
4. Chiudere lo sportello dell'unità dischi.

5. Posizionare su ON l'interruttore di tensione.

6. A questo punto il controllo passa al Microsoft BASIC II che si carica automaticamente nella memoria del calcolatore.

☐ CONTROLLO

Si può ora stabilire che il Microsoft BASIC II è stato caricato correttamente quando sullo schermo televisivo compaiono le seguenti informazioni



SENZA UNITA' DISCHI CON UNITA' DISCHI

Se non compare uno dei suddetti messaggi, occorre spegnere il calcolatore e ripetere il caricamento del Microsoft BASIC II.

LA COPIA DEL DISCHETTO DI ESTENSIONE DEL MICROSOFT BASIC II

Il dischetto di ESTENSIONE del Microsoft BASIC II contiene vari files che aggiungono ulteriori comandi e funzioni al programma Microsoft BASIC II. Tali files sono: DOS.SYS, DUP.SYS, AUTORUN. SYS, RS232.SYS, CIOUSR e MEM.SAV. Per evitare di distruggere questi importanti files, il dischetto di Estensione viene fornito in un formato "protetto contro la scrittura" (senza la tacca), il che significa che non si possono salvare programmi sul dischetto originale. Per questa ragione, bisogna fare una copia del dischetto di Estensione prima di iniziare a scrivere i propri programmi.
Utilizzare lasequente procedura per preparare copia di lavoro del dischetto di Estensione del Microsoft BASIC II:

1. Con il dischetto di Estensione inserito nell'unità a dischi, il Microsoft BASIC II caricato in memoria e sullo schermo la richiesta (-) di inserimento dati, battere DOS e premere RETURN. Una volta terminato il caricamento del DOS, che durera circa 30 secondi, lo schermo televisivo visualizzera il menu DOS.

2. Togliere il dischetto di Estensione dell'unità a dischi.

3. Inserire un dischetto da formattare nell'unità a dischi.

4. Qui di seguito vengono indicate la richiesta del calcolatore e le risposte dell'utente:

   CALCOLATORE : SELECT ITEM OR RETURN FOR MENU
   (Fai una selezione o torna al menu)

   UTENTE : I RETURN

   CALCOLATORE : WHICH DRIVE TO FORMAT?
   (Quale unita vuoi formattare?)

   UTENTE : 1 RETURN

   CALCOLATORE : TYPE "Y" TO FORMAT DISK 1
   (Batti "Y" per formattare disco 1)

   UTENTE : Y RETURN

5. L'unita a dischi entra in azione e funziona per meno di un minuto. Il processo di formattazione è finito quando sullo schermo appare di nuovo il messaggio: "SELECT ITEM OR RETURN FOR MENU".

   Una volta che il dischetto e stato formattato, il dischetto di Estensione puo essere duplicato. Premere RETURN per ottenere il menu DOS. Poi inserire il dischetto di Estensione nell'unità a dischi. Per eseguire la copia di lavoro agire come segue:

   UTENTE : J RETURN

   CALCOLATORE : DUP DISK-SOURCE, DEST DRIVES
   (Inserisci unita sorgente, unità di destinazione)

UTENTE : 1.1 RETURN

CALCOLATORE : INSERT SOURCE DISK, TYPE RETURN
(Inserisci il disco sorgente, batti RETURN)

UTENTE : RETURN

CALCOLATORE : TYPE "Y" IF OK TO USE PROGRAM AREA
CAUTION: A "Y" INVALIDATES MEM.SAV
(Batti "Y" se sei pronto ad usare l'area PROGRAMMA
ATTENZIONE: UN "Y" INVALIDA L'USO DI MEM.SAV)

(MEM.SAV è un file che salva il programma BASIC II ogni volta che si ritorna al menu DOS. Se non si utilizza tale file, si libera una parte di memoria che viene così resa disponibile per la duplicazione di dischetti).

UTENTE : Y RETURN

CALCOLATORE : INSERT DESTINATION DISK, TYPE RETURN
(Inserisci il dischetto di destinazione,batti RETURN)

(Togliere il dischetto di Estensione ed inserire il dischetto precedentemente formattato).

UTENTE : RETURN

CALCOLATORE : INSERT SOURCE DISK, TYPE RETURN
(Inserisci il dischetto sorgente, batti RETURN)

(Togliere il dischetto formattato ed inserire il dischetto di Estensione)

UTENTE : RETURN

CALCOLATORE : INSERT DESTINATION DISK, TYPE RETURN
(Inserisci il dischetto di destinazione, batti RETURN)

(Togliere il dischetto di ampliamento ed inserire di nuovo il dischetto formattato).

UTENTE : RETURN

CALCOLATORE : SELECT ITEM OR RETURN FOR MENU
(Fai una selezione o batti RETURN per ritornare al menu)

Il processo di duplicazione del dischetto di Estensione richiede di commutare due volte il dischetto nell'unità dischi. Quando il calcolatore emette il messaggio "SELECT ITEM OR RETURN FOR MENU", la copiatura è completata.

Si possono controllare ambedue le directories dei dischetti per verificare che tutto si sia svolto bene, nel seguente modo: Battere A e premere RETURN, poi premere di nuovo RETURN. Sullo schermo televisivo appare l'elenco dei files contenuti nel primo dischetto. Ese-

guire lo stesso controllo sull'altro dischetto inserito nell' unità. Per ulteriori informazioni consultare i manuali "Introduzione al Sistema Operativo per Disco (DOS)" o "Manuale di Riferimento del Sistema Operativo Disco II Atari".

6. Se si corrompe il file MEM.SAV, si perdono le istruzioni Basic per il disco. Per ripristinarle, è necessario ricaricare il Microsoft BASIC II. Posizionare l'interruttore di alimentazione prima su OFF e poi di nuovo su ON.

7. Quando sullo schermo televisivo appare il messaggio "SELECT ITEM OR RETURN FOR MENU", si può lasciare il menu DOS e tornare al Microsoft BASIC II battendo B (opzione "RUN CARTRIDGE" del menu DOS). Una volta reinserito il Microsoft BASIC II apparirà la richiesta (·) di inserimento dati. Il Microsoft BASIC II è ora pronto a ricevere i comandi dell'utente.

N O T A

Quando si esegue l'opzione "RUN CARTRIDGE" è importante che il dischetto da cui è stato caricato il DOS si trovi nell'unità dischi. Il sistema può bloccarsi se si usa un altro dischetto.

I PUNTI DI PARTENZA

Questo manuale descrive i numerosi vantaggi di programmazione offerti dal Microsoft BASIC II ATARI. Contiene tutte le informazioni necessarie per iniziare a sviluppare programmi semplici o complessi utilizzando il Microsoft BASIC II.

- Modi diretto o differito

Il Microsoft BASIC II accetta comandi sia in modo diretto che in modo differito - cioè i comandi possono essere inseriti ed eseguiti direttamente oppure preceduti da numeri di riga per creare programmi che diventeranno operativi solo dopo l'emissione del comando RUN. Il Microsoft BASIC II accetta numeri di riga da 0 a 63999.
La richiesta (·) sullo schermo televisivo significa che il calcolatore è pronto ad accettare comandi provenienti dall'utente. Quando si batte un comando, esso inizia ad apparire dove è posizionato il cursore, esattamente sotto il segno di inserimento dati. Si può battere un comando direttamente e premere il tasto RETURN per avere risultati immediati:

UTENTE: PRINT "CIAO, SONO IL TUO NUOVO BASIC II" RETURN
CALCOLATORE: CIAO, SONO IL TUO NUOVO BASIC II

Oppure si può battere un numero di riga ed iniziare la programmazione in modo differito:

UTENTE: 10 PRINT " MI PIACERAI" RETURN

In modo differito, quando si preme R RN non succede niente; il calcolatore inserisce le informazioni in oria e l'esecuzione vera e propria viene differita finchè non si b e RUN e si preme RETURN.

UTENTE : RUN RETURN
CALCOLATORE : MI PIACERAI

Poichè il comando RUN non è precedut a un numero di riga, viene eseguito direttamente ed avvia il progr a a partire dal primo numero di riga.

● Parole riservate (Keywords)

Un calcolatore esegue tutti i comanc critti con sintassi corretta. Il linguaggio di programmazione Micr ft BASIC II usa come comandi parole Inglesi. Tali parole vengono ch ate "Parole riservate" o "Parole chiave". Una parola chiave come ' NT" ordina al calcolatore di scrivere sullo schermo televisivo. alcolatore riconosce tali parole chiave come parole speciali e sa e trattarle. Nel vocabolario del Microsoft BASIC II ci sono più 00 parole riservate. Se si scrive una parola chiave in modo errato e ne usa una non riconosciuta dal calcolatore, il Microsoft BASIC stampa un messaggio di errore. Le parole chiave non possono essere te da sole come nomi di variabili in un programma, ma possono esse usate come parti di un nome di variabile. Per esempio, IF e GOSUB parole chiave e non possono essere usate come variabili, mentre E e RGOSUB sono ammesse. Si può trovare un elenco completo delle pa e chiave nell'Appendice N.

● La riga di programma del Microsoft IC II

Ogni riga di programma del Microsof ASIC II è formata da un numero di riga seguito da uno statement es sso in BASIC. I numeri di riga aiutano il Microsoft BASIC II a man ere la sequenza dei comandi, per eseguirli nell'ordine giusto.

```
# Riga    Statement
100       IF A=B THEN PRINT "EQUAL" SE PRINT "NOT EQUAL"
```

● Regole di punteggiatura

Come esistono segni di punteggiatur uando si scrive un qualunque testo, ne esistono anche nel Microsof BASIC II. Le regole di punteggiatura dipendono dai particolari requ ti dei comandi del Microsoft BASIC II.
Una regola generale richiede che tu i comandi siano in maiuscolo. Altre regole riguardano gli spazi t i comandi, i loro parametri e l'uso di virgolette, virgole, due p i, punto e virgola ed altri segni di punteggiatura.

● Spazi

Il Microsoft BASIC II ha una sola r la riguardante l'uso degli spazi nei programmi. Ogni parola chiave c e essere preceduta e seguita da u

no spazio. Comunque, ci sono casi in cui lo spazio non è obbligatorio. Per esempio, quando un delimitatore (come due virgolette) segue il comando ed è parte integrale di esso, lo spazio è opzionale.
Come regola generale, comunque, i programmi vanno scritti come frasi normali, con uno spazio prima e dopo ogni parola chiave.

● Virgolette

Il Microsoft BASIC II richiede il segno delle virgolette per indicare dove iniziano e dove terminano le stringhe di caratteri; come nella scrittura normale le virgolette definiscono l'inizio e la fine di un discorso diretto. Le virgolette indicano al calcolatore dove inizia e dove finisce una stampa.

Le doppie virgolette permettono l'uso di virgolette all'interno di una frase da stampare.

Esempio di programma:

```
UTENTE 100 PRINT " INIZIA UNA STAMPA SUL VIDEO .... ORA FERMATI"
UTENTE 110 PRINT " "" INIZIA ANCORA .... FERMATI""
UTENTE RUN RETURN
CALCOLATORE INIZIA UNA STAMPA SUL VIDEO.....ORA FERMATI
            "INIZIA ANCORA.....FERMATI
```

Da ora in poi non verranno più fornite le indicazioni "UTENTE" e "CALCOLATORE".

● Virgole

La virgola ha tre usi:

1) può essere usata per separare le opzioni richieste dopo una parola chiave. La parola chiave SOUND ha cinque diverse funzioni nel Microsoft BASIC II ATARI. Ogni parametro è separato da virgole. Per esempio, SOUND 2,&79,10,8,60 significa voce 2, frequenza esadecimale 79 (mezzoC), rumore 10, volume 8 e durata (in sessantesimi di secondo) 60, cioè un secondo.

   Un altro esempio di uso della virgola è lo statement SET COLOR 4,4,10 che significa registro 4, rosa, luminescenza brillante. La virgola indica dove finisce un'informazione e dove inizia la successiva. Il BASIC si aspetta di trovare i parametri di un comando nell'ordine esatto, separati da virgole.

2) la virgola può essere usata per separare valori opzionali e nomi di variabili. Con lo statement INPUT si può introdurre in una singola riga qualsiasi numero di nomi di variabili. Se ne possono usare quanti se ne vuole purchè siano separati da virgole.
   Per esempio, INPUT A,B,C,D,E indica al calcolatore che devono essere inseriti cinque valori da tastiera.

3) la virgola può essere usata per avanzare alla successiva colonna in uno statement PRINT.
Quando la virgola viene usata dopo una virgoletta o tra espressioni, farà avanzare la stampa alla successiva colonna che è un multiplo di 14 spazi. Per esempio, se ad X viene assegnato il valore 25, allora lo statement 10 PRINT " TU HAI ", X, " ANNI " produce la seguente spaziatura al momento dell'esecuzione. Ricordare che lo schermo televisivo può contenere solo 2 colonne in larghezza, per cui la seconda riga apparirà come nell'esempio:

```
|              |              |
|< 14 spazi >|< 14 spazi >|
|              |              |

TU  HAI        25
ANNI
```

● Punto e virgola

Il punto e virgola viene usato nello statement PRINT. Il punto e virgola introduce uno spazio dopo variabili e costanti, uno spazio vuoto (blank) iniziale prima dei numeri positivi ed il segno del meno, senza lasciare spazi vuoti, prima di numeri negativi. Per esempio, se ad X viene assegnato il valore 25, allora lo statement 10 PRINT "TU HAI"; X; " ANNI" quando viene eseguito produce la seguente spaziatura:

```
TU HAI 25 ANNI
```

Se ad X viene assegnato il valore -25, allora lo statement10 PRINT "TU HAI"; X; "ANNI" avrà la seguente spaziatura, quando viene eseguito il programma :

```
TU HAI-25-ANNI
```

Se si vuole che prima e dopo 25 ci sia più di uno spazio, bisogna lasciare lo spazio nella stringa all'interno di virgolette. Perciò :

```
10 PRINT "TU HAI  "; 25; "  ANNI"
```

fornirà la seguente spaziatura quando viene eseguito il programma:

```
TU HAI   25   ANNI
```

Il punto e virgola può essere usato anche per scrivere due statement PRINT, costanti di stringa, o variabili sulla stessa riga dello schermo televisivo.
Per esempio :

```
100 PRINT "IL VALORE E' £";
120 VALORE = 20
130 PRINT VALORE
UTENTE      : RUN RETURN
CALCOLATORE: IL VALORE E' £ 20
```

● Due punti

Il segno di due punti viene usato per concatenare più statement espressi su di una sola riga aventi un unico numero di riga, permettendo quindi l'esecuzione di molti statement eseguiti con lo stesso numero di riga. La tecnica di collegare più di uno statement su una sola riga, fa sì che il programma richieda meno memoria. Inoltre, si possono usare fino a tre righe di schermo o poco meno di 120 caratteri per ogni riga numerata.

Per esempio: 10 X=5:Y=3:Z=X+Y:PRINT Z:END
L'espediente dei due punti è spesso utilizzato dai programmatori per meglio organizzare il programma. Lo stesso programma con numeri di riga invece di due punti occupa più bytes di memoria:

```
10 X = 5
20 Y = 3
30 Z = X + Y
40 PRINT Z
50 END
```

● Editing

La tastiera del calcolatore Atari Modello XL ha i tasti e le funzioni differenti da quelle di una normale macchina da scrivere. La prima differenza è che i caratteri standard sono le maiuscole e per introdurre lettere minuscole bisogna premere il tasto CAPS LOWR. In questo modo operativo, la tastiera funziona come una normale macchina da scrivere, con il tasto SHIFT che esegue le maiuscole. Poichè la maggior parte dei programmi in BASIC vengono scritti in lettere maiuscole, il modo operativo più comune sarà appunto il maiuscolo. Per rientrare nel maiuscolo da CAPS LOWR, premere il tasto SHIFT e premere contemporaneamente CAPS LOWR.

● Tasti "Control" e "Shift"

I tasti di controllo del cursore permettono funzioni di editing immediate. Tali tasti vengono usati assieme ai tasti SHIFT e CTRL. I tasti che offrono speciali funzioni di editing sono descritti nei paragrafi seguenti.

Tenere il tasto di controllo CTRL premuto, mentre si premono i tasti direzionali (frecce), per spostare il cursore in qualsiasi punto dello schermo senza modificare nessuno dei caratteri già presenti sullo schermo stesso. Per quanto riguarda i tasti che hanno tre funzioni, premendo uno di essi assieme al tasto CTRL si produce il simbolo in alto a sinistra.

CTRL ↑	Sposta il cursore verso l'alto di una posizione senza modificare il programma o lo schermo
CTRL →	Sposta il cursore di uno spazio a destra senza modificare il programma o lo schermo
CTRL ↓	Sposta il cursore di una riga verso il basso senza modificare il programma o lo schermo
CTRL ←	Sposta il cursore di uno spazio verso sinistra senza modificare il programma o lo schermo

Per le precedenti quattro funzioni, se il cursore è su un limite dello schermo, spostandolo fuori da tale limite, esso riappare sul lato opposto dello schermo stesso.

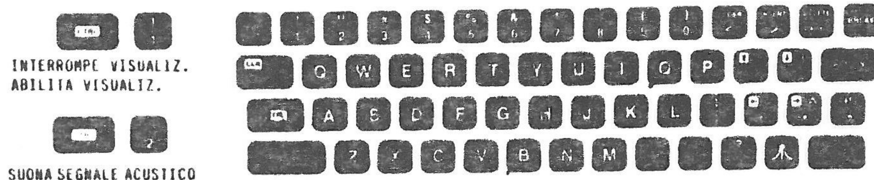
CTRL INSERT	Inserisce uno spazio di un carattere
CTRL DELETE BACK S	Cancella un carattere o uno spazio
CTRL 1	Arresta temporaneamente o riavvia la visualizzazione sullo schermo. Può essere usato durante la lista o l'esecuzione di un programma
CTRL 2	Fa suonare il segnale acustico del calcolatore

Tenendo il tasto SHIFT premuto mentre si premono i tasti numerici, si ottiene la visualizzazione dei simboli indicati nella parte superiore di tali tasti.

SHIFT INSERT	Inserisce una riga
SHIFT DELETE BACK S	Cancella una riga
SHIFT CAPS LOWR	Riporta lo schermo ai caratteri alfabetici maiuscoli.

- Caratteri grafici

Il tasto di controllo CTRL funziona come un secondo tipo di "SHIFT". Quando viene premuto assieme ad un altro tasto, sullo schermo appare un carattere proveniente da un set di caratteri completamente nuovo. Tali caratteri grafici possono essere usati per produrre interessanti disegni e grafici, con o senza la cartuccia ATARI BASIC.
Il diagramma illustrato qui di seguito mostra i caratteri grafici prodotti da ogni CTRL più una combinazione di tasti.



- Tasti con funzioni speciali

  Tasto "invers video". Premere questo tasto per invertire la luminosità di un carattere rispetto al colore di fondo. Premere il tasto una seconda volta per tornare al testo normale (testo luminoso su fondo scuro).

□ CAPS LOWR

Tasto delle minuscole. Premere questo tasto per cambiare i caratteri dello schermo da maiuscoli in minuscoli. Per riportare i caratteri in maiuscolo, premere il tasto SHIFT e il tasto CAPS LOWR simultaneamente.

□ ESC

Tasto di "escape". Premere questo tasto per inserire un comando di editing del video da eseguirsi più tardi (modo differito). Nel modo diretto, ripulisce il video se si preme simultaneamente CTRL e CLEAR. Quando in modo differito, volendo ripulire lo schermo al raggiungimento della riga 10, introdurre quanto segue:

10 PRINT "ESC CTRL CLEAR" e premere RETURN.

Il microsoft BASIC II permette inoltre di utilizzare il comando CLS nei modi diretto e differito per ripulire il video.

ESC viene usato assieme ad altri tasti per stampare speciali caratteri grafici.

□ BREAK

Tasto di interruzione. Questo tasto arresta l'esecuzione o la stampa del programma, visualizza un > sul video e posiziona il cursore sotto di esso. L'esecuzione può essere ripresa battendo CONT e premendo RETURN.

□ SYSTEM RESET

Tasto di "RESET" di sistema. Questo tasto arresta l'esecuzione del programma, riporta il funzionamento del video al modo grafico 0, pulisce lo schermo e riporta a tutti i valori standard.

□ CLR SET TAB

Tasto di tabulazione. Per impostare una tabulazione premere simultaneamente i tasti SHIFT e CLR SET TAB. Per cancellare una tabulazione, premere simultaneamente i tasti CTRL e CLR SET TAB. Usato da solo il tasto CLR SET TAB fa avanzare il cursore alla successiva posizione di tabulazione. In modo differito, l'utente imposta ed azzera una tabulazione abbinandola al comando PRINT preceduto da un numero di riga e seguito da un segno di virgolette, quindi premendo il comando ESC.

Esempio.

100 PRINT "ESC SHIFT CLR SET TAB

200 PRINT "ESC CTRL CLR SET TAB

Se non vengono fissate tabulazioni, esse sono inserite come standard alle colonne 7, 15, 23, 31 e 39.

□ RETURN

Tasto di "RETURN". Questo tasto viene usato per terminare un comando o uno statement BASIC. Premere questo tasto dopo ogni comando in modo diretto o dopo aver inserito una riga di programma.

2 Elementi del Programma

☐ COSTANTI E VARIABILI

Le costanti sono numeri o lettere usate in un programma e vengono mantenute sempre uguali in tutto il programma. Esempi di costanti sono : 5, "JACK".

Le variabili sono nomi assegnati a numeri o lettere ed il loro contenuto può cambiare durante lo svolgimento di un programma. Esempio di variabili sono : A, J$.

Ci sono cinque tipi di costanti e variabili nel Microsoft BASIC II ATARI: intero, reale a semplice precisione, reale a doppia precisione, esadecimale e stringa

● Creazione di un nome di variabile

I caratteri ammessi in un nome di variabile sono le lettere alfabetiche da A a Z, i numeri da 0 a 9 e il segno di sottolineatura ( ).
Il carattere di sottolineatura è un carattere valido nel Microsoft BASIC II ATARI. Nel nome di una variabile sono ammessi anche numeri ma esso deve iniziare con un carattere alfabetico. Il nome di variabile X9 è corretto, mentre 9X è considerato errato.

● Indicazione della precisione di variabili numeriche

Il tipo di variabile si può indicare in due modi:

1) Prestabilire la lettera iniziale di una variabile usando DEFINT, DEFSNG, DEFDBL o DEFSTR.
2) Identificare la variabile con un indicatore del tipo (%, # , $)

Il vantaggio di prestabilire il tipo di variabile consiste nel fatto che l'utente può modificare tutte le variabili da un tipo all' altro senza dover riesaminare tutto il programma per cambiare i nomi di variabile.Per esempio, se si modifica DEFINT A in DEFDBL A, tutte le variabili che iniziano con la lettera A vengono modificate dal tipo intero al tipo a precisione doppia. La seconda possibilità è quella di usare un identificatore di tipo : # (doppia precisione) % (intero), e $ (stringa). Gli identificatori vengono posti alla fine del nome di variabile stessa. Se una variabile è caratterizzata sia dall'identificazione DEF, sia dall'identificatore (# ,% , $), il secondo prevale sul primo.
Sebbene DEFSNG, DEFDBL , DEFINT e DEFSTR possono essere inseriti in qualunque punto del programma, essi vengono normalmente posti nella parte iniziale.

- **Costanti intere**

Esempi : 23, -9999, 709, 32000

Tutti i numeri interi del Microsoft BASIC II compresi tra -32768 e +32767 sono memorizzati in due bytes binari. Se una costante intera viene moltiplicata con un numero reale a semplice precisione, il prodotto è un numero reale a semplice precisione. I risultati di operazioni matematiche sono sempre memorizzati nel tipo di precisione di livello più alto.
I numeri interi negativi sono memorizzati come binari in complemento a due.

- **Variabili intere**

Esempi : SMALLNO% , J%, COUNT%

Un numero intero può essere identificato inserendo un segno di percentuale (%) come ultimo carattere nel nome della variabile. Un esempio di una variabile intera identificato dal nome, è NO%. (Il numero intero di 16 bit è memorizzato come binario di complemento a due).

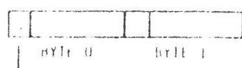
- **DEFINT**

Formato : DEFINT lettera, | lettera iniziale - lettera finale|
Esempi : 10 DEFINT N, J, K-M
          20 DEFINT I

N O T A

> Le righe verticali nel formato indicano una parte opzionale dello statement.
> Questa convenzione è adottata in tutto il manuale.

Le lettere iniziali di nomi di variabili identificate dallo statement DEFINT sono interi. Le variabili di interi aumentano la velocità di elaborazione, ma possono mantenere in modo corretto solo i valori compresi tra -32768 e +32767. Va ricordato che gli identificatori posti alla fine del nome di variabile sono prioritari. Sebbene N sia definito da DEFINT come un tipo di intero, # che appare dopo la N la identifica come a doppia precisione. Perciò N # , NI # e NUMB # sono tutte variabili numeriche a doppia precisione poichè il segno # significa doppia precisione.



bit di segno
0 è negativo
1 è positivo

Fig. 2-1 Rappresentazione macchina di variabili interi

● Costanti reali a semplice precisione

Esempi : 65E12, 333335, .45E8, .33E-6

Tutte le costanti di un programma all'esterno dell'intervallo -32768 a 32767 sono numeri reali in semplice precisione.

● Variabili reali a semplice precisione

Esempi : AMT, LENGTH ; BUFFER

Se la precisione di una variabile non viene dichiarata, tale variabile viene considerata dall'elaboratore automaticamente a singola precisione. I numeri memorizzati in semplice precisione hanno una accuratezza di 6 cifre significative. L'intervallo esponenziale va da -38 a +38.

● DEFSNG

Formato : DEFSNG lettera, | lettera iniziale - lettera finale |
Esempi : 100 DEFSNG K, S, A-F
120 DEFSNG Y

I nomi di variabili che iniziano con le lettere identificate in DEFSNG sono variabili reali a semplice precisione. In DEFSNG K, S, A-F, l'intervallo di lettere A-F significa che A, B,C,D,E,F, sono tutti a semplice precisione. I nomi di variabili che iniziano con K e S sono anch'esse a semplice precisione. Le lettere singole e i gruppi di lettere devono essere separati da virgole.

Esempio :

```
10 DEFSNG A-F
20 COUNTER = COUNTER +1
30 PRINT COUNTER
40 GOTO 20
```

Nell'esempio, tutti i nomi di variabili che iniziano con la lettera C sono a singola precisione. Perciò COUNTER è a semplice precisione perchè inizia con C. Se il contatore si chiamasse COUNTER # (# significa precisione doppia) esso avrebbe una doppia precisione anche se è stato definito come a semplice precisione. Ricordare sempre che gli identificatori nel nome della variabile sono prioritari.

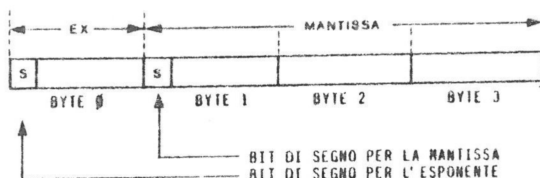


Fig. 2-2 Rappresentazione macchina di numero reale

● Costanti reali a doppia precisione

Esempi : 45D5, 23D-6, 8888888D-11

Si possono indicare i numeri reali a doppia precisione nelle costanti mettendo la lettera D prima della parte esponenziale. I numeri reali a doppia precisione sono memorizzati in 8 bytes ed hanno una precisione di 16 cifre decimali.

● Variabili reali a precisione doppia

Esempi : DBL #, X #, LGNO #

Il segno # identifica le variabili reali a doppia precisione. Una variabile reale a doppia precisione occupa 8 bytes. L'esponente e il segno sono memorizzati nel primo byte. L'intervallo esponenziale è lo stesso della semplice precisione: -38 a +38. L'accuratezza è di 16 cifre significative nei reali a doppia precisione. Il segno # viene posto dopo il nome della variabile.

● DEFDBL

Formato : DEFDBL lettera,| lettera iniziale - lettera finale|
Esempi : 10 DEFDBL Z, C-E
20 DEFDBL R

I nomi di variabile che iniziano con lettere identificate dallo statement DEFDBL sono reali a doppia precisione. Nel precedente esempio CDE, Z e R sono dichiarate a doppia precisione. Il nome di variabile E1 è una variabile a doppia precisione poichè inizia con E.
La figura 2-3 illustra come vengono rappresentati in memoria i numeri reali a precisione doppia.

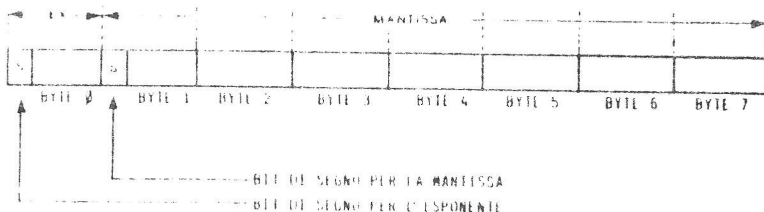


Figura 2-3 Rappresentazione macchina di variabile a doppia precisione

● Costanti esadecimali

Esempi : &76, &F3, &7B, &F3EB

E' spesso più semplice indicare indirizzi e codici in linguaggio macchina in notazione esadecimale (base 16) piuttosto che in notazione decimale. Se un numero e preceduto dal simbolo &, significa che è esadecimale.

Per richiamare la routine in linguaggio macchina che inizia all'indirizzo esadecimale C305, bisogna specificare A = USR (&C305.0). A = PEEK (&5A61) assegnerà il contenuto dell'indirizzo di memoria 5A61 alla variabile A. La notazione esadecimale è utile nella grafica specialmente utilizzando "players" e "missiles". Segue una tabella di equivalenza per i numeri decimali, esadecimali e binari.

Decimale	Esadecimale	Binario
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

Tab. 2.1 Equivalenze decimali, esadecimali e binarie

☐ STRINGHE E MATRICI

● Costanti di stringa

Esempio: "AMMONTARE", "IMMETTI IL NOME ____________"

Le costanti stringa sono sempre racchiuse tra virgolette. La costante stringa può avere qualsiasi lunghezza fino alla lunghezza massima di riga (120). Le stringhe sono composte da tutti i caratteri presenti sulla tastiera. "!-$%&&"())OOKJHGGFDS". All'interno di una stringa si può anche usare il segno di doppia virgoletta. La doppia virgoletta si trasformerà in un segno di virgolette semplici quando la stringa viene stampata. La barra verticale (|) ed il carattere null(Ø) vengono usati internamente per indicare la fine di una stringa. Usando uno di tali caratteri in una stringa si troncherà la stringa a quella posizione. Segue un esempio di una costante stringa usata in uno statement di stampa:

```
10 PRINT "Stringhe e %&'$"" cose """;
20 A$ = "STRINGA DI COSTANTI ASSEGNATA AD UNA VARIABILE"
30 PRINT A$
RUN RETURN
```

Tale programma stamperà : Stringhe e % &' $ "cose".
STRINGA DI COSTANTI ASSEGNATA AD UNA VARIABILE

- Variabili stringa

Esempi : A$, NINT$, ADDRESS$

I nomi di variabili stringa terminano con un segno $. Una variabile stringa può essere assegnata ad una stringa lunga al massimo come una riga. Il segno di doppie virgolette ("") è un modo per ottenere all'interno di una stringa un segno di virgolette semplici.
Esempi di stringhe assegnate a A$:

```
10 A$ = "una stringa"
20 A$ - "un'altra""stringa"""
```

- DEFSIR

Formato : DEFSIR lettera, | lettera iniziale - lettera finale |
Esempi : 10 DEFSIR A, K-M, Z
         20 DEFSIR F, J, I, O

Un nome di variabile può essere definito come una stringa indicando la sua lettera iniziale nello statement DEFSTR. Le stringhe possono essere lunghe al massimo come una riga. Come in tutte le dichiarazioni di nomi di variabile, l'identificatore in fondo al nome è prioritario.
A# e A% sono i tipi di identificatore (rispettivamente a doppia precisione e per un numero intero) anche se la loro lettera iniziale è definita da DEFSIR.
Esempio di programma

```
10 DEFSIR A, M, Z
20 A = "Nome Impiegato VALORE"
30 PRINT A
```

Il programma stamperà l'intestazione : Nome Impiegato VALORE

- Matrici

Una matrice è un elenco di variabili indicizzate, con lo stesso nome, come A (0), A (1), A (2). La gamma degli indici va da zero al valore dimensionato. La figura 2-4 illustra una matrice di 7 elementi.

A (0)
A (1)
A (2)
A (3)
A (4)
A (5)
A (6)

Fig. 2-4 Esempio di matrice

E' ammesso l'uso di un indice di valore massimo uguale a 10 in una lista o in una matrice senza dover usare lo statement di dimensionamento (DIM). Con il valore standard di OPTION BASE che è ZERO (0), si può creare una matrice di 11 elementi senza la necessità di dimensionamento.

Per esempio, per una matrice chiamata UNA_MATRICE

```
100 UNA_MATRICE (1) = 55
120 UNA_MATRICE (2) = 77
130 UNA_MATRICE (3) = 93
140 UNA_MATRICE (4) = 61
150 FOR X = 1 TO 4
160 PRINT UNA_MATRICE (X)
170 NEXT
180 END
```

Matrici a più dimensioni sono un insieme di più matrici ad una dimensione. Per esempio, una matrice a due dimensioni contiene due colonne. Le righe vanno in orizzontale e le colonne in verticale.

Matrici multi-dimensionali vengono memorizzate dal BASIC in ordine di riga crescente. Ciò significa che tutti gli elementi della prima riga sono memorizzati per primi, seguiti da tutti gli elementi della seconda riga e così via. La figura 2-5 illustra una matrice 7 X 4.

RIGHE	COLONNE			
	M(0,0)	M(0,1)	M(0,2)	M(0,3)
	M(1,0)	M(1,1)	M(1,2)	M(1,3)
	M(2,0)	M(2,1)	M(2,2)	M(2,3)
	M(3,0)	M(3,1)	M(3,2)	M(3,3)
	M(4,0)	M(4,1)	M(4,2)	M(4,3)
	M(5,0)	M(5,1)	M(5,2)	M(5,3)
	M(6,0)	M(6,1)	M(6,2)	M(6,3)

Fig. 2-5 Esempio di matrice multidimensionale

☐ OPERATORI ARITMETICI DI RELAZIONE E LOGICI

● Operatori aritmetici

Gli operatori aritmetici sono : (), =, -, ^ ,*, /, +, - (il primo trattino significa negazione, l'ultimo trattino significa sottrazione).

Quando si creano espressioni i simboli aritmetici possono essere usati insieme ad operatori logici. L'espressione A/C > D*A è valida. Le espressioni aritmetiche rappresentano simboli matematici. Il simbolo * indica l'operazione di moltiplicazione. Il simbolo ^ viene usato da Mi

crosoft BASIC II Atari per indicare l'esponente. L'ordine di precedenza è il seguente :

SIMBOLO	SIGNIFICATO
( )	L'espressione tra le parentesi viene elaborata per prima
=	Segno di uguale
-	Numero negativo. Non è una sottrazione ma un segno negativo davanti ad un numero. Esempi : -3, -A, -6
^	Esponente
*	Moltiplicazione
/	Divisione
+	Addizione
-	Sottrazione

● Operatori di relazione

Gli operatori relazionali vengono valutati da sinistra a destra.

OPERATORE	SIGNIFICATO
=	Uguale. Questo e il vero uso del segno di uguale. Esso domanda se A = B. Il B non e assegnato ad A.
<> o ><	Non uguale. Valuta se due espressioni non sono uguali.
<	Minore di. A minore di B è rappresentato da A < B.
>	Maggiore di. A maggiore di B è rappresentato da A > B.
>= o =>	Maggiore di o uguale a. A maggiore di o uguale a B è rappresentato da A >= B.
<= o =<	Minore di o uguale a. A minore di o uguale a B è rappresentato da A <= B.

Gli operatori di relazione (=,<>,<,>,<=,>=) possono eseguire delle attività molto utili sulle stringhe. L'ordine alfabetico può essere ottenuto velocemente mediante un algoritmo che usi l'espressione A$<B$. Si può eseguire un confronto tra due nomi chiedendosi se A$ = B$. Le variabili di stringa vengono valutate come numeri in codice ATASCII (per esempio, la lettera A è 65 mentre B è 66, così A < B è sempre vero).

A$<B$ Vero (non zero) se A$ ha un numero di codice ATASCII minore di B$

Esempio di ordinamento

```
100 INPUT A$, B$
120 IF A$<B$ THEN 160
130 C$ = A$
140 A$ = B$
150 B$ = C$
160 PRINT A$, B$
170 END
```

Per fare una prova, battere delle combinazioni di due parole e separarle da virgole. Le parole saranno selezionate in ordine alfabetico usando il suddetto esempio. Così si vedrà che BILL precede BILLY e CANE precede GATTO.

- Operatori logici

Gli operatori logici hanno il seguente ordine di priorità

OPERATORE	SIGNIFICATO
NOT	Gli 8 bit del numero vengono complementati. Se è un binario, dopo questa operazione diventa 0.
AND	I bits del numero sono sottoposti alla operazione logica di AND. Esempio: A AND B. Se A è 1 e B è 1, il risultato è 1. Se A è 1 e B è 0, il risultato è 0. Se A è 0 e B è 0, il risultato è 0.
OR	I bits del numero sono sottoposti alla operazione logica di OR. Esempio A OR B. Se A è 1 e B è 1, il risultato è 1. Se A è 1 e B è 0 il risultato è 1. Se A è 0 e B è 1, il risultato è 1. Se A è 0 e B è 0, il risultato è 0.
XOR	I bits del numero sono sottoposti alla operazione logica di XOR (OR esclusivo). Esempio: A XOR B. Se A è 1 e B è 1, il risultato è 0. Se A è 1 e B è 0, il risultato è 1. Se A è 0 e B è 1, allora il risultato è 1. Se A è 0 e B è 0 il risultato è 0.

Gli operatori logici possono essere usati con variabili di stringa (A$). Consultare la Sezione "Funzioni di Stringa".

NOTA

Gli operatori di relazione e gli operatori logici possono essere combinati insieme per formare delle espressioni. Gli operatori di relazione sono prioritari rispetto agli operatori logici.

Per esempio, A > B AND C < D è un'espressione. I simboli maggiore di e minore di sono considerati per primi, poi viene valutato AND. Se la relazione è vera, il risultato è un numero diverso da zero. Se la relazione non è vera, il risultato è zero. Non zero è vero, zero è falso.

In uno statement IF, questa valutazione determina che cosa accadrà successivamente. Vengono eseguiti l'ELSE o il successivo numero di riga quando l'espressione formata con operatori è falsa.

3 Comandi del Programma

Questa sezione descrive quei comandi generalmente usati in modo diretto.

☐ AUTO

(Disponibile solo con il dischetto di estensione).

Formato : AUTO [n, i]
Esempi : AUTO 200, 20
AUTO

AUTO numera automaticamente le righe. Se non si specifica "n, i" (numero iniziale, incremento) i numeri di riga inizieranno da 100 con incremento 10. Il comando AUTO deve essere usato quando si inizia a scrivere un programma. Introdurre AUTO, un numero di riga iniziale, una virgola ed il valore del quale si vuole che i numeri di riga aumentino. Premere poi RETURN per avviare la numerazione AUTO. Un nuovo numero di riga verrà automaticamente stampato dopo aver introdotto uno statement e premuto RETURN. Per arrestare AUTO, premere RETURN da solo senza battere uno statement. AUTO può essere arrestato anche premendo il tasto BREAK.

<u>Programma di esempio</u>

AUTO 300, 20 RETURN La numerazione inizia da 300, con incremento 20.

```
300 PRINT "CIO' MOSTRA COME"
320 PRINT "LA NUMERAZIONE AUTOMATICA"
340 PRINT "FUNZIONA"
360 RETURN
■
```

N O T A

Se al nuovo numero di riga da generare c'è già una riga, tale riga verrà visualizzata sullo schermo televisivo.

☐ CLOAD

Formato : CLOAD
Esempi : CLOAD
440 CLOAD

CLOAD carica un programma da una cassetta nella memoria RAM prima dell'esecuzione. Quando si introduce CLOAD e si preme RETURN, viene generato un segnale acustico.

Bisogna quindi posizionare il nastro all'inizio del programma usando il contatore del nastro come guida, e premere PLAY sul registratore Atari. Premere di nuovo il tasto RETURN. Istruzioni per caricare un programma con CLOAD sono contenute nel Manuale dell'operatore del Registratore.

[] CONT

Formato : CONT
Esempio : CONT

CONT riprende l'esecuzione di un programma dal punto in cui era stato interrotto da uno STOP, dal tasto BREAK o da un errore di programma. Questa istruzione è molto utile quando si deve eseguire il "debugging" (ricerca e correzione errori) di un programma. Si può stabilire un punto di arresto mediante lo statement STOP. Le variabili possono essere controllate al punto in cui l'istruzione si arresta usando :PRINT nome di variabile in modo diretto (senza un numero di riga). Poi il programma può essere ripreso usando lo statement CONT.

[] CSAVE

Formato : CSAVE
Esempi : CSAVE
330 CSAVE

CSAVE salva su cassetta un programma residente in memoria RAM. CSAVE salva la versione codificata del programma. Immettendo CSAVE e premendo RETURN l'indicatore acustico incorporato suona due volte per indicare che bisogna premere PLAY e RECORD sul registratore. Poi premere di nuovo RETURN. Tali tasti non devono essere premuti finchè non è stato posizionato il nastro.

Salvare un programma con questo comando è più veloce che farlo usando SAVE"C:" perchè vengono utilizzati spazi più brevi tra record e record. Si può usare SAVE"C:" con LOAD"C:" o CSAVE con CLOAD ma non si devono usare questi statements accoppiati diversamente. Infatti SAVE"C:" con CLOAD darà un messaggio di errore.

[] DEL
(Disponibile solo con il dischetto di Estensione)

Formato : DEL n-m
Esempi : DEL 450-
DEL 250-350
DEL -250

DEL cancella gli statement di programma che trova in memoria. Con il comando DEL si può cancellare un solo statement o più statements alla volta. Un trattino viene usato per indicare l'intervallo di statements

da cancellare :

DEL n : Cancella solo uno statement ("n" è un numero di statement)

DEL-m : La cancellazione inizia dal primo statement del programma e termina con lo statement "m". Anche lo statement "m" viene cancellato.

DEL n- : La cancellazione inizia con il numero di statement "n" e prosegue fino all'ultimo statement del programma.

DEL n-m : La cancellazione inizia con "n" e termina con "m". Vengono cancellati anche ambedue gli statements "n" e "m".

Esempio di programma :
```
100 PRINT "UN ESEMPIO DI COME"
120 PRINT "FUNZIONA IL COMANDO"
130 PRINT "DELETE"
DEL 120- RETURN
```

In memoria viene lasciato solo lo statement 100.
```
LIST RETURN
100 PRINT "UN ESEMPIO DI COME"
```

Se volete cancellare un solo statement da un programma, basta battere il numero dello statement e premere RETURN.
Programma di esempio :
```
110 FOR X = 1 TO 5000:NEXT
110 RETURN
```

N O T A

Se si tenta di usare DEL in modo differito, il programma si arresta dopo la cancellazione dei numeri di riga.

□ DOS

Formato : DOS
Esempio : DOS

Il comando DOS permette di abbandonare il BASIC e di ottenere il Menu del Sistema Operativo a Disco. Ciò rende disponibili tutte le scelte del menu DOS sui programmi e sui dati memorizzati su dischetti.Per tornare al Microsoft BASIC II Atari, selezionare l'opzione B nel menu DOS. Il richiamare Menu DOS cancella il programma BASIC in memoria a meno che non si abbia sul dischetto un file MEM.SAV. (Vedere il Manuale del Sistema Operativo Dischi - ATARI).

□ KILL

Formato : KILL "device:program_nome"
Esempio : KILL"D:PROG1.BAS"

KILL cancella il programma specificato da un'unità.

□ LIST

Formato : LIST|"device:program_nome"||m-n|
Esempio : 100 LIST
150 LIST "C:
120 LIST "P:" 10-40
100 LIST "D:GRAFX.BAS
110 LIST 100-200
100 LIST -300

List produce un elenco degli statement del programma, attualmente in memoria, sullo schermo televisivo o su un'altra unità. Se è presente "device:program-nome", lo statement di programma attualmente in memoria viene scritto sull'unità indicata.
Nomi validi di unità sono : D: (per dischi), C: (per cassette),P: (per stampante). Se LIST viene usato senza un nome di unità, il sistema utilizza lo schermo (E:). Il nome di un programma può essere un qualsiasi nome lungo fino a 8 caratteri con una estensione-nome di tre caratteri.

Quando si listano programmi sullo schermo, è spesso utile fermare la visualizzazione del listing mentre sta rollando (scrolling) premendo simultaneamente CTRL e 1. Per far riprendere lo scrolling del listing premere di nuovo CTRL e 1 contemporaneamente.

Con il comando LIST si possono listare una o più righe di programma.Un trattino (-) viene usato per indicare l'intervallo di statements:

LIST — Lista l'intero programma, dal numero di riga più basso al più alto.
LIST n — Lista solo lo statement "n" (dove "n" è un numero di statement).
LIST-m — Il listing inizia con il primo statement del programma e si arresta dopo aver listato lo statement "m".
LIST n- — Il listing inizia con il numero di statement "n" e continua fino all'ultimo numero di statement del programma.
LIST n-m — Il listing inizia con lo statement "n" e finisce con lo statement "m". Ambedue tali statement sono compresi nel listing.

Programma di esempio : (da notare che REM indica un commento e non un comando eseguibile. Vedere REM nella Sezione 4).

110 REM Esempio di listing

```
120 PRINT "MOSTRA QUALE STATEMENT"
130 PRINT "O GRUPPO DI STATEMENT"
140 PRINT "SIANO LISTATI".
```

LIST 110-130 RETURN

Il Microsoft BASIC II visualizza quanto segue :

```
110 REM Esempio di listing
120 PRINT "MOSTRA QUALE STATEMENT"
130 PRINT "O GRUPPO DI STATEMENT"
```

Esempio di LIST usato in modo differito:

```
10 COUNT = 1
20 COUNT = COUNT + 1
30 PRINT COUNT
40 IF COUNT <> 30 THEN 20
50 LIST
```

L'utilizzo di LIST per listare un programma su stampante viene eseguito in modo diretto, introducendo LIST "P:"

L'utilizzo di LIST per listare un programma in forma ASCII non codificata su dischetto, viene eseguito introducendo LIST "D:nome.ext".

Usare LOAD per ricaricare programmi non codificati nella memoria del calcolatore. LOAD può essere usato per introdurre programmi che sono stati listati o salvati su cassetta o dischetto.

☐ LOAD

Formato : LOAD"Unità:nome_programma"
Esempi : LOAD"D:EXAMPLE"
110 LOAD"C:"

LOAD "UNITA':nome_programma" sostituisce il programma in memoria con uno che si trova nell'unità indicata.

. Come "unità" si può indicare un'unità a disco o cassetta.

. Usare LOAD "C:" per caricare dati o files presenti su cassetta.

. Per i programmi su cassetta, che sono stati precedentemente salvati con CSAVE, usare CLOAD.

. Per i files su dischetti, usare "D:nome_programma" per i programmi listati o salvati su di essi. (Vedere anche MERGE).

☐ LOCK

Formato : LOCK"unità:nome_file".

Esempio : LOCK "D:CHECKBK"

LOCK offre una protezione contro cancellazioni accidentali di files. Una volta che un file è bloccato, non può essere riscritto, cancellato o rinominato.

[] MERGE

Formato : MERGE "unità:nome_programma"
Esempio : MERGE "D:STOCK.BAS"

Usare MERGE per fondere un programma memorizzato su di un'unità con il programma che si trova nella memoria della macchina.

Se si incontrano numeri di riga doppi, la riga del programma presente sull'unità specificata sostituisce quella del programma presente in memoria.

Alla fine dell'operazione di fusione, viene emesso un messaggio di errore 136 (fine-file, EOF - END OF FILE).

Programma di esempio : (vedere spiegazione di REM alla Sezione 4)

```
100 REM QUESTO PROGRAMMA
120 REM CHE SI FONDE
130 REM PROGRAMMA
  LIST "D:STOCK.BAS"
110 REM E' UN ESEMPIO
115 REM PER MOSTRARE UN PROGRAMMA
125 REM CON UN ALTRO
  MERGE "D:STOCK.BAS"
  LIST
100 REM QUESTO PROGRAMMA
110 REM E' UN ESEMPIO
115 REM PER MOSTRARE UN PROGRAMMA
120 REM CHE SI FONDE
125 REM CON UN ALTRO
130 REM PROGRAMMA
```

[] NAME... TO

(Disponibile solo con il dischetto di Estensione)

Formato : NAME "unità:nome_programma_1" TO "nome_programma_2"
Esempio : NAME "D:BALANCE" TO "CHECKBK"

NAME attribuisce un nome nuovo ad un file. L'unità deve essere fornita solo per il programma vecchio, mentre dopo la parola TO deve essere indicato solo il nuovo nome del programma, chiuso tra virgolette.

☐ NEW

Formato : NEW
Esempio : NEW
100 IF CODE <> 642 THEN NEW

NEW cancella il programma presenta in memoria e permette di introdurre un nuovo programma. Il comando NEW non distrugge il tempo memorizzato con il comando TIME$.

Quando si esegue NEW, tutte le variabili vengono azzerate e tutte le stringhe annullate.

☐ RENUM

(Disponibile solo con il dischetto di Estensione)

Formato : RENUM |m, n, i|
Esempio : RENUM 10, 100, 10

m = nuovo numero di riga da assegnare al primo statement rinumerato.
n = primo numero di riga da cui partire a rinumerare.
i = incremento tra i nuovi numeri di riga generati.

RENUM assegna nuovi numeri di riga a determinate righe di un programma. I valori standard di RENUM sono 10, 0, 10.
RENUM modifica tutti i riferimenti a righe contenuti negli statements GOTO, GOSUB, THEN, ON ... GOTO, ON ... GOSUB e ERROR per assumere i nuovi numeri di riga.

N O T A

RENUM non può essere usato per modificare l'ordine delle righe di programma. Per esempio, RENUM 15,30 non sarebbe permesso se il programma avesse tre righe con numeri 10, 20 e 30. Non possono essere creati numeri di riga maggiori di 63999.

Esempi :

RENUM	Rinumera l'intero programma. Il primo numero di riga sarà 10. L'incremento tra le righe sarà di 10.
RENUM 10, 100	Il vecchio numero di riga 100 verrà rinumerato a 10. L'incremento sarà di 10 (lo standard è 10).
RENUM 800,900,20	Rinumera le righe da 900 alla fine del programma.La riga 900 diventa 800. L'incremento è di 20.
RENUM 300,140,20	Assegna il numero 300 alla riga 140. L'incremento è di 20.

PRIMA	DOPO
100	100
110	110
120	120
130	130
140	300
150	320
160	340
170	360

□ RUN

Formato : RUN|"unità:nome_programma"||opzionale numero riga di inizio|
Esempi : RUN
RUN, 120
200 RUN "D:TEST.BAS"
110 RUN 200

RUN senza un numero di riga avvia l'esecuzione del programma dallo statement con il numero di riga più basso.

RUN inizializza tutte le variabili numeriche a zero ed annulla le variabili di stringa, prima di eseguire il primo statement del programma.

RUN può essere usato in modo differito (con un numero di riga). Può essere usato anche per elaborare un programma da dischetto o da cassetta. Comunque, quando RUN viene usato per eseguire un programma presente su dischetto o cassetta (per esempio, RUN "D:SHAPES"), esso non può essere usato con "numero di riga_opzionale" in quanto questa opzione può essere usata solo per eseguire programmi che sono già in memoria.

RUN può essere usato solo per eseguire programmi salvati con l'istruzione SAVE.
Esempio : RUN 250
Programma di esempio : 200 RUN "D:TEST"

Quando viene eseguito il numero di riga 200 dello statement, verrà eseguito il programma chiamato TEST.

□ SAVE

Formato : SAVE "unità:nome_programma"
Esempio : SAVE "D:GAME.BAS"

SAVE copia il programma presente in memoria sul file indicato da "nome_programma".

Unità valide sono : D: (per dischi), C: (per cassette). Per esempio il comando SAVE "D:TEMP.BAS" salverà il programma attualmente presente in memoria su un dischetto. Il programma è registrato in forma codificata su nastro o dischetto.

Esempio :
SAVE "D:PROGRAM". Salva il programma in memoria sul file, su dischetto, con nome PROGRAM.
SAVE "C:". Salva il programma su cassetta (non è richiesto nessun nome di file).

N O T A

Un programma salvato con il nome AUTORUN.AMB è un programma di inizializzazione automatica, cioè verrà eseguito immediatamente all'accensione del sistema.

☐ SAVE . . . LOCK

Formato : SAVE "unità:nome_programma" LOCK
Esempio : SAVE "D:PROGRAM.EXA" LOCK

SAVE "unità:nome_programma" LOCK salva un programma su nastro o dischetto e lo codifica in modo tale che non possa essere editato, listato, fuso, esaminato o modificato. LOCK viene usato per salvaguardare il programma da eventuale furto o da manomissioni.

☐ TROFF
(Disponibile solo con il dischetto di estensione)

Formato : TROFF
Esempio : 770 TROFF

Questo comando disabilita il meccanismo di tracciamento utile nella ricerca di eventuali errori (vedi TRON). TROFF può essere usato in modo diretto o differito.

☐ TRON
(Disponibile solo con il dischetto di estensione)

Formato : TRON
Esempi : TRON
550 TRON

Questo comando abilita il meccanismo "Trace" per la ricerca di eventuali errori. Quando TRON è attivo, il numero di ogni riga incontrata vie

ne visualizzato sullo schermo televisivo prima di essere eseguita.TRON può essere usato in modo diretto o differito.

[] UNLOCK

Formato : UNLOCK "unità:nome_programma"
Esempio : UNLOCK "D:GAME1.BAS"

Lo statement UNLOCK ripristina un file in modo che possa essere riscritto, cancellato o ridenominato. Non è possibile sbloccare un file che è stato salvato (SAVE) con l'opzione LOCK.

[] VERIFY
(Disponibile solo con il dischetto di Estensione)

Formato : VERIFY "unità:nome_programma".
Esempi : VERIFY "D:BIO.BAS"
VERIFY "C:

VERIFY confronta il programma in memoria con quello indicato da :"unità:nome_programma". Se i due programmi non sono uguali insorge il messaggio di errore per mancata corrispondenza (FILE MISMATCH ERROR).

4 Statements

☐ AFTER

Formato : AFTER (tempo in 1/60 di secondo) |GOTO| numero di riga.
Esempio : 100 AFTER (266) GOTO 220

Quando viene eseguito lo statement AFTER, viene fatto un conteggio del tempo a partire da zero fino al numero indicato di sessantesimi di secondo. Quando l'intervallo di tempo è finito, l'esecuzione del programma riprende a partire dal numero di riga specificato.

AFTER può trovarsi in qualsiasi punto del programma, ma deve essere eseguito con lo scopo di iniziare un conteggio del tempo. E' ammesso un periodo massimo di tempo pari a 24 ore.
Quando si eseguono i comandi RUN, STOP o END, il contatore del tempo AFTER viene posizionato a 0.
Programma di esempio :

```
10 CLS:AFTER (300) GOTO 70
20 PRINT "HAI 5 SECONDI PER PREMERE UN TASTO,"PRINT"QUALUNQUE TASTO"
30 IF INKEY$ = ""THEN 30
40 PRINT "GRAZIE"
50 CLEAR STACK
60 END
70 PRINT "IL TEMPO E' FINITO!";
80 RESUME 50
```

☐ CLEAR

Formato : CLEAR
Esempio : CLEAR
          550 CLEAR

CLEAR azzera tutte le variabili e tutte le matrici ed annulla tutte le stringhe. Se dopo l'uso del comando CLEAR è necessario usare una matrice, essa deve essere ridimensionata.

☐ CLEAR STACK

Formato : CLEAR STACK

Esempio : 100 CLEAR STACK

CLEAR STACK è un modo per far abortire lo statement AFTER. Se in un programma si verificano certe condizioni, l'utente può voler cancellare lo statement AFTER.

Programma di esempio :

```
10 AFTER (120) GOTO 30
20 CLEAR STACK
25 STOP
30 PRINT "IL TUO TURNO E' FINITO"
40 STOP
```

☐ CLOSE

Formato : CLOSE #iocb
Esempio : CLOSE #2

CLOSE viene usato dopo che tutte le operazioni sui file sono terminate. Il segno # è obbligatorio ed il numero identifica l'IOCB (Blocco di controllo per l'Input/Output).

\# Simbolo obbligatorio
iocb Numero di un IOCB aperto precedentemente

☐ COMMON

Formato : COMMON Nome_variabile |nome_variabile|
COMMON ALL
Esempi : 110 COMMON I, J, A$, H%, DEC
100 COMMON ALL

Lo statement COMMON viene usato per mettere in comune valori di variabili utilizzate da più programmi. COMMON fa sì che le variabili di due programmi abbiano lo stesso nome e gli stessi valori. Se una variabile viene chiamata COUNT in un breve programma e tale programma viene collegato ad un altro che ha COUNT come variabile, il programma considera i COUNT come variabili diverse. Lo statement COMMON indica che si vuole che le due COUNT siano considerate come la stessa variabile. COMMON ALL mantiene uguali durante l'esecuzione valori di variabili.

Programma di esempio :

```
100 COMMON X
110 X = 4
120 RUN "D:PROG2"

BREAK
PRINT X RETURN
```

Il valore di X quando la riga 120 esegue PROG2 è 4.
Se esiste una variabile chiamata X in PROG2, X prende il suo valore dallo statement COMMON presente nel nuovo programma.

☐ DEF
(Disponibile solo con il dischetto di Estensione)

Formato : DEF nome_funzione (variabile|, variabile|) = definizione_funzione.
Esempio : 150 DEF MULT(J, K) = J*K

Una funzione definita dall'utente nella forma DEF A(X) = X ^ 2, dove A(X) rappresenta il quadrato di X, può essere usata in un programma come se fosse parte del linguaggio BASIC stesso. Normalmente una funzione definita dall'utente si trova nella parte inziale di un programma.
La funzione definita dall'utente non può occupare più di una riga di programma. Sono ammesse anche funzioni di stringa. Se la funzione definita è una variabile di stringa, l'espressione definita deve fornire un risultato di stringa. Possono essere definiti uno o più parametri.
Perciò DEF S$(A$, B$) = A$ + B$ è una definizione valida.

Programmi di esempio :

```
 5 !DEFINISCE LA FUNZIONE PER LA MEDIA
10 DEF AVG(X,Y) = (X + Y) /2
20 PRINT AVG(25,35)
30 END
RUN RETURN
30
```

Programma di esempio :

```
100 !DETERMINA LA POSIZIONE DELLA MANOPOLA
110 DEF PADDLE(X) = PEEK(624 + X)
120 PRINT PADDLE(0)
130 GOTO 120
100 !DETERMINA IL TASTO DEL COMANDO A CLOCHE
110 DEF STRIG(X) = PEEK(644 + X)
120 IF STRIGO(0) THEN 420 ELSE PRINT "BANG!"
130 GOTO 420
```

N O T A

DEF non può essere usato in modo diretto.

☐ DIM

Formato : DIM nome_variabile_aritmetica (numero_di_elementi),|list|

DIM nome_variabile_stringa (numero_di_elementi), |list|

Esempio : 10 DIM A$ (35), TOTAMT (50)

Lo statement DIM comunica al calcolatore il numero di elementi che l'utente vuole usare in una matrice.
Se si tenta di introdurre in una matrice più elementi di quanti ne siano stati previsti nello statement di dimensionamento, viene emesso un messaggio di errore.
La matrice più semplice è la matrice ad una sola dimensione. Si supponga di avere 26 studenti in una classe. Si può registrare un voto per ogni studente con il seguente dimensionamento :

```
10 OPTION BASE 1
20 DIM SCORE (26)
30 SCORE (1) = 55
40 SCORE (2) = 86
50 PRINT SCORE (1), SCORE (2)
RUN
```

Va notato che lo statement OPTION BASE pone a 1 il valore iniziale per gli indici; perciò SCORE (1) memorizza il punteggio numerico del primo studente. OPTION BASE 0 permette di iniziare con un indice uguale a Ø.
Il Microsoft BASIC II ATARI permette di avere fino a 255 dimensioni di matrici. Le matrici tridimensionali permettono di eseguire facilmente calcoli molto complessi :

```
110 X = 10:Y = 10:Z = 10
120 DIM BOXES(X, Y, Z)
10 REM 11 elementi nella matrice
20 DIM GROUP1 (10)
30 For I = 0 to 10: GROUP1 (I) = 1:PRINT GROUP1(I):NEXT
40 END
```

```
 5 REM 10 elementi nella matrice
10 OPTION BASE1
20 DIM GROUP2 (10)
30 FOR I = 1 TO 10:GROUP2(I):PRINT GROUP2(I):NEXT
40 END
```

[] END

Formato : END

Esempio : 990 END

END arresta l'esecuzione di un programma ed è normalmente l'ultimo statement di un programma. Quando END termina un programma, sullo schermo appare il carattere di richiesta comandi (prompt).

Nel Microsoft BASIC II ATARI non è necessario terminare un programma con lo statement END.

☐ ERROR

Formato : ERROR codice_errore
Esempio : 640 ERROR 162

ERROR seguito da un codice di errore costringe il BASIC a valutare un errore avente un determinato tipo di codice di errore. Forzare il verificarsi di un errore è una tecnica usata per verificare come il programma si comporta quando l'utente commette un errore. Nell'Appendice O viene fornito un elenco completo dei codici di errore.
Si possono usare sia errori di sistema che errori del BASIC.

☐ FOR ... TO ... STEP/NEXT

Formato : FOR variabile iniziale = valore iniziale TO valore finale STEP |incremento|valore
Esempio : 10 FOR X = 1 TO 500 STEP 3
150 FOR Y = 20 TO 12 STEP -2
30 FOR COUNTER = 1 TO 250

FOR e NEXT vengono usati insieme per eseguire ripetutamente un insieme di istruzioni, finchè una variabile raggiunge un determinato valore. La variabile parte con un valore iniziale, aumenta del valore dell'incremento ad ogni passo finché non raggiunge il valore finale.
FOR/NEXT determina quante volte possono essere eseguiti ripetutamente degli statement compresi tra i numeri di riga del FOR... TO... STEP e del NEXT.

Se STEP viene omesso, viene assunto il valore 1. STEP può essere un numero negativo o una frazione decimale.

Il seguente programma stampa 30 numeri, con le loro radici quadrate.

Programma di esempio :

```
100 FOR X = 1 TO 30
110 PRINT X, SQR (X)
120 NEXT
```

☐ GET

Formato : GET # iocb |AT (settore, byte); |nome_variabile
Esempi : 200 GET #1, X
330 GET #3,AT(8,2);J,K,L

GET legge un byte (valore da 0 a 255) dal file indicato dal #iocb e quindi memorizza tale byte nella variabile specificata.
Il programma di esempio richiede che esista un file chiamato "MIOFILE" sull'unità a dischi.
Prima di utilizzare il programma, usare il programma di esempio relativo allo statement PUT.

Programma di esempio

```
110 OPEN #1, "D:MIOFILE" INPUT
120 GET #1, A,B,C
130 CLOSE #1
140 PRINT A,B,C
```

N O T A

GET non può essere usato in modo immediato.

[] GOSUB/RETURN

Formato : GOSUB numero_di_riga
Esempio : 330 GOSUB 150

GOSUB provoca l'esecuzione della riga indicata dal "numero_di_riga". Uno statement RETUN segue la fine della subroutine e riporta l'esecuzione allo statement che segue lo statement GOSUB.

[] GOTO

Formato : GOTO numero_di_riga
Esempio : 10 GOTO 110

GOTO comunica al sistema il numero di riga da eseguire successivamente. Normalmente, gli statement vengono eseguiti in sequenza dal numero di riga più basso al più alto, ma GOTO può modificare tale sequenza. GOTO causa un salto nel programma al numero di riga indicato dopo GOTO.
Esempio : GOTO 55

Poiché questo statement non ha un numero di riga, esso avvia immediatamente l'esecuzione del programma presente in memoria a partire dalla riga 55.

```
100 PRINT "CIO' NON HA FINE"
120 GOTO 100
RUN RETURN
```

Questo programma provoca un salto continuo al numero di riga 100. Perciò lo schermo televisivo si riempie velocemente con CIO' NON HA FINE. Premere BREAK per arrestare il programma.

☐ IF... THEN

Formato : IF condizione THEN numero riga o statement
Esempi : 10 IF A = B THEN 290
20 IF J > Y AND J < V THEN PRINT "OUT OF STATE TAX"

Se il risultato della condizione verificata è vero, il successivo statement eseguito è quello indicato da "numero_di_riga". Un test viene eseguito con operatori matematici o di relazione. Il test può essere fatto su numeri o stringhe. Le parole GOTO dopo THEN sono opzionali. Se la condizione del test è falsa, l'esecuzione passa al successivo numero di riga del programma.

Programma di esempio :

```
160 IF NUMERO > ALTRO_NUMERO THEN 300
200 PRINT "ALTRO_NUMERO E' MAGGIORE"
250 STOP
300 PRINT "NUMERO E' MAGGIORE"
450 END
```

☐ IF... THEN...ELSE

Formato : IF condizione THEN andare al numero di riga o statement ELSE andare al numero di riga o statement.
Esempio : 250 IF R < Y THEN 450 ELSE 500

Questo statement è uguale allo statement "IF...THEN" con l'eccezione che l'esecuzione passa alla clausola ELSE se la condizione del test è falsa.

☐ INPUT

Formato : INPUT | #iocb ||" stringa di richiesta di inserimento"; |nome_variabile| nome_var|
Esempi : 120 INPUT "BATTI IL TUO NOME"; A$
350 INPUT "N.CONTO.,NOME";NUM, B$

INPUT permette all'utente di comunicare con un programma battendo informazioni sulla tastiera del calcolatore.

Con lo statement INPUT si possono anche stampare stringhe di caratteri. Ciò permette di scrivere richieste per l'utente come "BATTI IL TUO NOME". I caratteri battuti sono assegnati a nomi di variabili quando si preme il tasto RETURN o si batte una virgola. Lo statement INPUT arresta temporaneamente il programma finchè l'input da tastiera non è terminato. Lo statement INPUT genera automaticamente un punto interrogativo sullo schermo televisivo, per indicare la richiesta di inseri-

mento.

N O T A

Non sono ammesse virgole quando l'input avviene tramite tastiera. INPUT non è ammesso in modo diretto.

[ ] INPUT...AT

Formato : INPUT| #iocb, |AT (s,b) nome_di variabile
Esempio : 300 INPUT #5, AT (9,7) X

Se un'unità a dischi è stata aperta come INPUT ed ad essa è stato assegnato un IOCB # , allora essa può essere usata per introdurre dati. L'input dall'unità viene letto AT (settore, byte) e ad esso è assegnato un nome di variabile. INPUT #6, AT (x, y) X può essere usato per leggere una determinata posizione dello schermo.

[ ] LET

Formato : |LET| nome_variabile =| espressione_aritmetica| oppure | espressione_di stringa| nome_di variabile =|espressione_aritmetica| o | espressione_di stringa|
Esempi : 100 LET COUNTER = 55
120 D = 598

LET permette di assegnare un numero ad un nome di variabile. Il segno di uguale nello statement LET significa "assegnare" e non "uguale a" in senso matematico.

Per esempio, LET V = 9 assegna il valore 9 ad una variabile chiamata V. Il numero a destra del segno uguale può essere un'espressione composta di molti simboli matematici e nomi di variabili. Così, LET V = (X + Y - 9) / (W * Z) è uno statement valido.
La parola LET è opzionale. Tutto ciò che è necessario per l'assegnazione è il segno uguale.
Perciò,
100 LET THIS = NUMBER * 5
è equivalente a
100 THIS = NUMBER * 5

[ ] LINE INPUT

Formato : LINE INPUT | #iocb || "stringa di richiesta di inserimento"| nome di variabile stringa
Esempi : 190 LINE INPUT ANS $

Dalla tastiera viene introdotta una riga intera. Sono ammessi virgole, due punti, punto e virgola ed altri separatori. La fine della riga si indica premendo RETURN.

Programma di esempio :

```
100 LINE INPUT "QUAL E' IL TUO NOME?"; N$
120 PRINT N$
130 END
```

☐ LINE INPUT ... AT

Formato : LINE INPUT #iocb, AT (s.b) |"stringa di richiesta di inserimento"| nome di variabile|
Esempio : 300 LINE INPUT #5, AT (9, 7) X

Se un'unità dischi è stata aperta come LINE INPUT ed è stata assegnata ad un IOCB # essa può essere usata per introdurre dati. L'input dall'unità viene letto tramite AT (settore, byte) e viene assegnato a un nome di variabile. LINE INPUT #6, AT (x, y); X può essere usato per leggere una determinata posizione dello schermo.

☐ MOVE

Formato : MOVE da, a; numero di bytes
Esempio : 20 MOVE MADDR1, MADDR2, 9

Lo statement MOVE sposta i bytes di memoria da un'area di memoria all'altra. La prima posizione del blocco di memoria da spostare è data dalla prima espressione numerica (indirizzo da...) mentre la prima posizione del blocco di destinazione è data dalla seconda espressione numerica (indirizzo a...). La terza espressione numerica indica quanti bytes devono essere spostati. L'ordine di movimento è tale che il contenuto del blocco di dati non viene modificato dal movimento stesso. L'uso principale di MOVE è nei grafici del missile-giocatore.

Esempio : MOVE 55, 222, 5

Cinque bytes a partire dall'indirizzo 55 (cioè 55-60) verranno spostati nelle posizioni 222-226.

☐ NEXT

Formato : NEXT |nome_variabile|
Esempio : 30 NEXT J, I
          40 NEXT VB
         120 NEXT

NEXT trasferisce l'esecuzione al numero di riga FOR ... TO codificato precedentemente, finchè non viene superato il numero specificato dopo TO. Nel Microsoft BASIC II ATARI, NEXT non deve necessariamente essere seguito da un nome di variabile. Quando NEXT non è seguito da un nome di variabile, l'esecuzione viene trasferita a un precedente statement FOR... TO non ancora terminato.

Programma di esempio :

```
100 FOR X = 10 TO 100 STEP 10
110 PRINT X
120 NEXT
130 END
RUN RETURN
```

Il programma visualizza a video :

```
10
20
30
40
50
60
70
80
90
100
```

Due o più "variabili iniziali" possono essere combinate insieme nella stessa riga NEXT, separate da virgole.

Programma di esempio :

```
100 FOR X = 1 TO 20
110 FOR Y = 1 TO 20
120 FOR Z = 1 TO 20
130 NEXT Z, Y, X
```

☐ NOTE
(Disponibile solo con il dischetto di Estensione)

Formato : NOTE #iocb, nome_variabile, nome_variabile
Esempio : 120 NOTE 4, I, J

NOTE viene usato per memorizzare il numero del settore del dischetto su cui è posizionata la testina di lettura nel primo "nome di variabile" ed il valore del byte nella seconda variabile.

NOTE indica all'interno del file specificato le posizioni di lettura o scrittura del successivo byte.

☐ ON ERROR

Formato : ON ERROR |GOTO| numero di riga
Esempio : ON ERROR 550

In generale, l'esecuzione del programma si arresta quando incontra un errore e sul video appare un messaggio di errore. ON ERROR, a fronte dell'errore, provoca l'esecuzione delle istruzioni a partire dal numero di riga specificato.
Lo statement ON ERROR deve essere codificato prima che l'errore si verifichi realmente per poter trasferire l'esecuzione alla riga indicata. Per riprendere l'esecuzione normale bisogna usare lo statement RESUME, che fa ripartire il programma a partire dall'istruzione successiva a quella in cui si è verificato l'errore.
Quando vengono eseguite le istruzioni RUN, STOP o END, lo statement ON ERROR viene disattivato fino al successivo statement ON ERROR.

Programma di esempio :

```
10 ON ERROR 1000
20 PRINT  #3, "LINEA"
30 STOP
1000 PRINT "UNITA' NON ANCORA APERTA"
1010 STOP
1020 RESUME
```

Lo statement ON ERROR numero di riga può essere disabilitato dallo statement: ON ERROR GOTO 0. Se si disabilita ON ERROR all'interno della routine di trattamento dell'errore, l'errore viene processato in modo normale.

☐ ON ... GOSUB/RETURN

Formato : ON espressione_aritmetica GOSUB numero di riga_1, numero di riga_2, numero di riga_3...
Esempio : 220 ON X GOSUB 440, 500, 700

ON... GOSUB stabilisce quale subroutine debba essere eseguita successivamente(in base al valore dell'espressione aritmetica). Se il valore è 1, l'esecuzione passa "numero di riga_1". Se il valore è 2, l'esecuzione passa al "numero di riga_2"; se è 3, l'esecuzione passa al "numero di riga_3", e così via.

Programma di esempio :

```
100 INPUT "INSERISCI UN NUMERO (1-4)"; X
110 ON X GOSUB 130, 140, 150, 170
120 GOTO 100
130 PRINT  "PRIMA CHIAMATA - X = 1" : RETURN
140 PRINT  "SECONDA CHIAMATA - X = 2": RETURN
```

```
150 PRINT "TERZA CHIAMATA - X = 3" : RETURN
160 PRINT "NON PUOI ARRIVARE QUI", QUESTA RIGA NON APPARTIENE A GOSUB
170 PRINT "FINE PROGRAMMA" : END
```

[] ON...GOTO

Formato : ON espressione_aritmetica GOTO numero di riga_1, numero di riga_2, numero di riga_3.

Esempio : 400 ON X GOTO 550, 750, 990

ON ... GOTO stabilisce quale riga deve essere eseguita successivamente in base al numero rappresentato dall'espressione_aritmetica. Se tale numero è 1, il controllo passa al "numero di riga_1"; se è 2 passa al "numero di riga_2"; se è 3, passa al "numero di riga_3".

[] OPEN

Formato : OPEN #iocb,"unità:nome_programma" tipo di accesso_al file.

Esempi :
```
130 OPEN #4, "K:" INPUT
100 OPEN #3, "P:" OUTPUT
150 OPEN #4, "D:PROG.SAV" INPUT
120 OPEN #2, "D:GRAPH 1.BAS" UPDATE
110 OPEN #5, "D:PROG.BAS" APPEND
```

#	Carattere obbligatorio inserito dall'utente
iocb	Blocco di controllo di input/output (IOCB). Scegliere un numero da 1 a 7 per identificare un file ed il relativo accesso al file. Bisogna codificare il segno # seguito da un numero IOCB (1-7) e da una virgola. (Consultare il Manuale "ATARI Home Computer System Technical Reference Notes, per una dettagliata spiegazione dell'IOCB).
"unità:nome_programma"	Indica l'unità e il nome del programma. Le unità sono : D: (dischi), P: (stampante), E: (schermo), K: (tastiera), C: (cassetta), S: (schermo televisivo) e R: (RS 232-C). Quando viene usato D:, occorre far seguire il nome dal programma che può essere lungo fino a 8 caratteri e può avere un'estensione-nome di tre caratteri. I nomi di programma devono iniziare con un carattere alfabetico. All'inizio di questo capitolo si trova una descrizione completa dei codici di unità (K:, P:, C:, D:, E:, S:, R:).

accesso_al file	Indica il tipo di operazione:
	INPUT = operazione di input (ingresso)
	OUTPUT = operazione di output (uscita)
	UPDATE = operazione di input e output
	APPEND = permette di aggiungere registrazioni alla fine di un file.

Il concetto operativo collegato allo statement OPEN è quello di associare un numero (il numero IOCB) al nome di un file e alle sue caratteristiche di accesso. Dopo che in un programma si incontra lo statement OPEN # n, si possono usare PRINT # 2, INPUT # 3, NOTE # 5, STATUS # 2, GET #4 e PUT # 4; cioè si può usare il numero IOCB come un identificatore dal file. Gli statements OPEN # n e PRINT # n possono ora sostituire LPRINT (LINE PRINTING):

```
100 OPEN #3, "P:" OUTPUT
110 PRINT #3, "QUESTA E' UNA STAMPA DI PROVA"
120 CLOSE #3
```

I seguenti identificatori IOCB hanno utilizzi preassegnati.

- #0 è usato per l'INPUT e l'OUTPUT sull'unità E:, l'editore del video
- # 6 è usato per l'INPUT e l'OUTPUT sull'unità S: cioè in tutti i modi grafici

Un esempio dell'uso di IOCB # 6 è il seguente:

```
100 GRAPHICS 2
110 PRINT #6, AT (5, 5); "PROVA DI VISUALIZZAZIONE"
```

Gli IOCB da #1 e #5 (e IOCB # 7) possono essere usati liberamente,ma gli IOCB preassegnati non dovrebbero essere usati.

□ OPTION BASE

Formato : OPTION BASE 0 (Standard)
OPTION BASE 1

Esempi : 150 OPTION BASE1
200 DIM Z (25, 25, 25) ! Matrice con indici compresi tra 1 e 25.

OPTION BASE dichiara che il valore iniziale dell'indice può essere 0 o 1. L'OPTION BASE (0/1) dovrebbe essere il primo statement eseguibile di un programma. Se OPTION BASE viene omesso, il valore iniziale dell'indice e 0.

Programma di esempio :

```
100 REM ESEMPIO DI STATEMENT OPTION BASE 1.
110 OPTION BASE 1
```

```
120 DIM ARRAY (15, 15)
150 READ ARRAY (1,1), ARRAY (2,2), ARRAY (15,15)
165 DATA 32, 33, 34
180 PRINT ARRAY (1,1), ARRAY (2,2), ARRAY (15,15)
190 END
```

☐ OPTION CHR1, OPTION CHR2, OPTION CHRØ

Formati : OPTION CHR1
OPTION CHR2
OPTION CHRØ

Esempi : 110 OPTION CHR1
120 OPTION CHR2
130 OPTION CHRØ

OPTION CHR1 riserva 1024 bytes in memoria per i dati.

OPTION CHR2 riserva 512 bytes in memoria per i dati.

OPTION CHRO rilascia tutti i bytes riservati a OPTION CHR.

OPTION CHR1 e OPTION CHR2 vengono usati per riservare memoria per un insieme di caratteri nella memoria RAM. Il set di caratteri ROM può essere spostato (MOVE) nella nuova area RAM che è stata riservata e si può definire un set di caratteri completamente nuovo VARPTR (CHR1) o VARPTR (CHR2) puntano agli indirizzi iniziali. E' necessario tramite l'istruzione POKE (inserire un byte specificato in una posizione di memoria specificata) trasferire un nuovo indirizzo iniziale in CHBAS (vedere tabella E-2 nell'Appendice K). Ciò può essere fatto stabilendo la pagina a cui VARPTR (CHR1) o VARPTR (CHR2) puntano. Un modo per determinare e POKE (inserire un byte specificato in una posizione di memoria specificata) un nuovo CHBAS è:

```
300 CHBAS = & 2F4
310 ADDR% = VARPTR (CHR1)
320 POKE CHBAS,((ADDR%/256) AND &FF)
```

L'istruzione GRAPHICS (vedere Sezione 6) deve sempre precedere lo statement OPTION CHRn, poichè l'elaboratore deve sempre conoscere il modo grafico prima che l'utente possa riservare spazio.
Questa procedura maschera il byte (MSB) più significativo dell'indirizzo di memoria VAPTR e trasferisce (POKE) tale MSB in CHBAS così che l'utente può eseguire la commutazione al nuovo set di caratteri. Vedere l'Appendice C per un esempio di redifinizione dell'insieme di caratteri.

☐ OPTION PLM1, OPTION PLM2, OPTION PLMØ

Formati : OPTION PLM1
OPTION PLM2
OPTION PLMØ

Esempi : 100 OPTION PLM1
100 OPTION PLM2
100 OPTION PLMØ

OPTION PLM1 riserva 1280 bytes di memoria per il gioco giocatore e missile (risoluzione a riga singola). OPTION PLM2 riserva 640 bytes di memoria per il gioco giocatore e missile (risoluzione a riga doppia). OPTION PLMO rilascia gli spazi riservati dalle altre OPTION PLM. L'istruzione GRAPHICS (vedere Sezione 6) deve sempre precedere lo statement OPTION PLMn poichè il calcolatore deve conoscere il modo grafico prima che l'utente possa riservare spazio.
OPTION PLM1 o OPTION PLM2 vengono usati per riservare memoria per il gioco giocatore e missile, per azzerare la memoria e per posizionare PMBASE (vedere Tabella E-2 nell'Appendice E). L'utente non deve preoccuparsi per quanto riguarda l'allocazione di una adeguata area di memoria per il gioco giocatore e missile quando si usano gli statements OPTION PLM. Per trovare l'esatta posizione di memoria del byte iniziale del missile, occorre iniziare VARPTR (PLM1) o VARPTR (PLM2).
Bisogna memorizzare (POKE) nella locazione decimale 53277 il valore decimale 3 per abilitare i grafici del giocatore e missile. Si deve anche trasferire (POKE) nella locazione decimale 559 il valore decimale 62 per soluzioni a riga singola o il decimale 46 per soluzioni a riga doppia.

Vedere la Sezione 7 per un esempio di grafici del giocatore e missile.

☐ OPTION RESERVE

Formato : OPTION RESERVE n

Esempio : 300 OPTION RESERVE 24

Nello statement OPTION RESERVE n , "n" rappresenta il numero di bytes riservati. Per esempio OPTION RESERVE 24 riserva 24 bytes. VARPTR (RESERVE) può essere usato per indicare l'indirizzo di inizio dei 24 bytes in OPTION RESERVE 24. Questo statement permette di riservare bytes per codici macchina o per altri scopi.

☐ PRINT

Formati : PRINT "costante_di stringa"

```
? "costante_di stringa", nome_variabile
PRINT nome_variabile_1, nome_variabile_2,...nome_variabile_n
```

Esempi .

```
100 PRINT "PROGRAMMA DI ORDINAMENTO"; A$, X
500 ? #6, "STAI ENTRANDO NELLA SEGRETA"!Stampa per GRAPHICS
1 e 2
```

PRINT visualizza sullo schermo televisivo costanti di stringa, variabili di stringa o variabili numeriche. Lo statement PRINT lascia una riga vuota quando viene eseguito da solo (senza parametri). Il simbolo di punto interrogativo (?) ha lo stesso significato della parola PRINT.

Programma di esempio :

```
100 PRINT "SALTA UNA RIGA"
120 PRINT
125 REM   "NOTA L'USO DI""PER STAMPARE UNA VIRGOLETTA"
130 ANOTHER_LINE$ = "STAMPA ""UN'ALTRA"" RIGA"
140 ? ANOTHER_LINE$
150 END
```

La riga 120 lascia una riga vuota quando viene eseguito questo programma :

```
RUN RETURN
SALTA UNA RIGA
STAMPA "UN'ALTRA" RIGA
```

Costanti di stringa, variabili di stringa e variabili numeriche vengono stampate tutte sulla stessa riga quando la costruzione della riga comprende una virgola o un punto e virgola. Non è necessario usare una virgoletta di chiusura se si vuole visualizzare una costante di stringa sullo schermo televisivo :

```
100 PRINT "QUI NON C'E' LA VIRGOLETTA DI CHIUSURA
RUN RETURN
QUI NON C'E' LA VIRGOLETTA DI CHIUSURA
```

[ ] PRINT... AT

Formati : PRINT #iocb, AT(A,T)x,y
PRINT #6, AT(x,y) "costante_di stringa";nome_variabile

PRINT... AT stamperà un determinato settore e byte se l'unità dischi è stata aperta come OUTPUT (vedere statement OPEN). La clausola AT è molto versatile. Se l'unità da indirizzare è un'unità dischi, AT (s,b) si riferisce al settore e al byte. Mentre se l'unità da indirizzare è lo schermo, come in PRINT o PRINT #6, AT(x,y) si riferisce alla posizione dello schermo x,y.

Esempio di stampa su un'unità dischi :

```
100 OPEN#3,"D:TEST.DAT" OUTPUT
110 X=5
120 PRINT#3, AT(7,1)"TEST";X
130 CLOSE#3
```

N O T A

La posizione di Settore e byte deve essere stata assegnata al file aperto in precedenza prima di poter scrivere su di esso.

Esempio di visualizzazione su una posizione dello schermo:

```
100 GRAPHICS 1
110 PRINT#6, AT(3,3)"VISUALIZZA SULLO SCHERMO"
```

☐ PRINT ... SPC

Formato : SPC (n)

Esempio : 10 PRINT TAB (5);"XYZ";SPC(7)"7 SPAZI A DESTRA DI XYZ"

SPC inserisce spazi tra variabili e costanti in una riga da stampare. SPC conta gli spazi iniziando dalla posizione in cui è stato stampato l'ultimo carattere.

☐ PRINT... TAB

Formato : TAB (n)

Esempio : 120 PRINT TAB (5); "LA STAMPA INIZIA 5 SPAZI DOPO"

TAB sposta il cursore del numero di posizioni indicate tra parentesi. Questo statement viene usato con PRINT per spostare i caratteri dopo un numero specificato di spazi.

TAB conta sempre gli spazi iniziando dalla prima posizione del margine sinistro.

Programma di esempio :

```
100 PRINT "QUESTA RIGA INIZIA A TAB (0)"
110 PRINT TAB (5); "QUESTA RIGA INIZIA A TAB (5)"
120 END
```

☐ PRINT USING
(Disponibile solo con il dischetto di Estensione)

PRINT USING permette all'utente di formattare un output in vari modi:

- le cifre di variabili numeriche possono essere posizionate esattamente dove si vuole
- si può inserire un punto decimale negli importi espressi in valuta
- si può inserire un segno di dollaro ($) immediatamente prima del primo digit di un importo in valuta
- si può inserire un segno di dollaro di fronte ad un importo
- gli importi possono essere riempiti a sinistra con asterischi (***$45.00) per protezione
- i numeri possono essere convertiti in formato esponenziale (E) o a doppia precisione (D)
- un segno di più (+) provoca la stampa di un + per i numeri positivi e di un - per i numeri negativi.

(1) PRINT USING #

Il segno # mantiene una posizione per ogni cifra di un numero. Si possono inserire digit a destra o a sinistra del punto decimale usando il segno # . Vengono inseriti degli zeri a destra del decimale, se necessario, nel caso in cui l'importo in valuta sia un intero. Quando si usa # , i punti decimali vengono automaticamente allineati. Tale segno è molto utile nella programmazione finanziaria.

Programma di esempio :

```
10 X = 246
20 PRINT USING "###";X
RUN RETURN
    246
```

N O T A

Se un numero ha più cifre del numero di specificati, di fronte al numero viene stampato un segno di percentuale.

Programma di esempio :

```
110 X = 99999
110 PRINT USING "###";X
120 END

RUN RETURN
% 99999
```

(2) PRINT USING.

Si può inserire il punto in qualsiasi posizione all'interno di una

stringa di indicatori #. Il decimale nell'importo si allineerà con il decimale specificato nella USING.

```
10 X = 2.468
20 PRINT USING "##.##";X
```

```
RUN RETURN
2,47
```

N O T A

Poichè dopo il punto decimale sono state indicate solo due cifre, la posizione dei centesimi viene arrotondata automaticamente.

(3) PRINT USING,

Si può posizionare una virgola in qualsiasi posizione di cifra PRINT USING. Il simbolo di virgola fa sì che venga stampata una virgola a sinistra di ogni gruppo di tre cifre nel risultato.

N O T A

Più indicatori di posizione decimale (#) devono essere usati se nel risultato sarà inserita più di una virgola.

Programma di esempio :

```
 5 DEFDBL X
10 X = 2933604.53
20 PRINT USING "########.##";X
30 END
RUN RETURN
2,933,604.53
```

(4) PRINT USING **

Due asterischi nelle prime due posizioni riempiono gli spazi non utilizzati del risultato con asterischi. I due asterischi sono come due cifre aggiuntive.

Programma di esempio :

```
100 X = 259
120 PRINT USING "** #######.##";X
RUN RETURN
****** 259.00
```

(5) PRINT USING $

Un segno di dollaro in posizione iniziale fa sì che un segno di dollaro venga stampato a sinistra del risultato.

Programma di esempio :

```
100 X = 3.59631
110 PRINT USING "$###.##";X
120 END
```

```
RUN RETURN
$    3.60
```

(6) PRINT USING $$

Un doppio segno di dollaro ($$) nelle prime due posizioni dà nel risultato un segno di dollaro mobile. Cioè il segno di dollaro verrà posizionato immediatamente vicino alla prima cifra decimale visualizzata.

Programma di esempio :

```
100 X = 3.5961
110 PRINT USING "$$###.##";X
120 END
```

```
RUN RETURN
$ 3.60
```

(7) PRINT USING **$

Se **$ viene usato nelle prime tre posizioni, il risultato conterrà degli asterischi che riempiono le posizioni non utilizzate e un segno di dollaro verrà inserito nella posizione immediatamente precedente al primo digit visualizzato.

Programma di esempio:

```
100 X = 53.29
110 PRINT USING "**$ #######.##";X
120 END
```

```
RUN RETURN
********$ 53.29
```

(8) PRINT USING ∧∧∧∧

Con quattro simboli esponenziali dopo l'indicatore di posizione decimale (#), il risultato viene fornito in forma esponenziale (E o D).

Programma di esempio :

```
100 X = 500
110 PRINT USING " ## ∧∧∧∧ ";X
120 END
```

```
RUN RETURN
5E+02
```

(9) PRINT USING +

Il segno più (+) stampa un + davanti ai numeri positivi e un - davanti a i numeri negativi. Il segno più (+) può essere usato in testa o in coda alla stringa PRINT USING.

Programma di esempio :

```
100 A = 999.55
110 PRINT USING "+ ####";A
120 END
```

```
RUN RETURN
+ 1000
```

(10) PRINT USING -

Il segno meno (-) dopo la stringa PRINT USING fa apparire un - dopo un numero negativo. Se il numero è positivo apparirà uno spazio.

Programma di esempio :

```
100 A = -998
110 PRINT USING "----"; A
120 END
```

```
RUN RETURN
998-
```

(11) PRINT USING !

Il segno di esclamazione (!) estrae il primo carattere da una stringa.

Programma di esempio :

```
100 A$ = "B MATEMATICA 1A"
110 PRINT USING "!"; A$
120 END
```

```
RUN RETURN
B
```

(12) PRINT USING %bbbb%

Il segno di percentuale (%) e spazi vuoti (b) estraggono parte di una stringa da una stringa più lunga. La lunghezza della stringa da estrarre è 2 più il numero di spazi (b) compresi tra i due segni di percentuale.

Programma di esempio :

```
100 A $ = "GARUTI PINO"
120 PRINT USING "%bbbb%", A$
130 END
```

```
RUN RETURN
GARUTI
```

☐ PUT

Formati : PUT # iocb, | AT (settore, byte); |espressione_aritmetica.

Esempi : 100 PUT # 6, AT (8,2); J, K, L

GET e PUT sono due statement opposti. PUT memorizza il valore di un singolo byte, compreso tra 0 e 255, sul file indicato da # iocb (# è un carattere obbligatorio per ambedue i comandi).

Il programma di esempio, fornito qui di seguito, crea un file chiamato "MIOFILE" e memorizza tre numeri in questo file su dischetto. Si usi il seguente programma di esempio fornito per lo statement GET (Vedi) per ottenere i suddetti tre numeri.

Programma di esempio :

```
10 OPEN #1,"D:MIOFILE" OUTPUT
20 PUT #1, 65, 66, 67
30 CLOSE #1
```

☐ RANDOMIZE

Formato : RANDOMIZE |seed|

Esempi : 10 RANDOMIZE
10 RANDOMIZE 55! Stabilisce una determinata sequenza ripetibile

RANDOMIZE assicura che si verifichi una diversa sequenza casuale di numeri ogni volta che viene eseguito un programma con la funzione aritmetica RND (vedere la Sezione 5)
RANDOMIZE determina un seme casuale per la sequenza RND.

Programma di esempio :

```
100 RANDOMIZE
110 PRINT RND
120 END
```

Ogni volta che si esegue il suddetto programma, sullo schermo televisivo viene visualizzato un numero singolo. Senza il comando RANDOMIZE, la funzione aritmetica RND ripete lo stesso numero pseudo-casuale ogni volta che viene eseguito un programma. Quando si controlla un programma può essere utile avere una sequenza RND che si ripeta ad ogni esecuzione. In questo caso la funzione RND deve essere usata da sola, senza RANDOMIZE.

Un altro modo per produrre una lunga sequenza che sia la stessa ogni volta, è quello di usare RANDOMIZE |seme| (dove |seme| è un numero arbitrario) invece, se per esempio si vuole vedere un diverso grup

po di carte ogni qualvolta si esegue un gioco, occorre usare RANDOMIZE da solo nella parte iniziale del programma.

Esempio di RND senza RANDOMIZE :

```
100 PRINT RND
110 END
```

Ogni volta che viene eseguito questo programma, sullo schermo televisivo appare lo stesso numero.

□ READ/DATA

Formato : READ nome_variabile_1, |nome_variabile_2,||
nome_variabile_n|

Esempio : 150 READ A, B

READ assegna numeri o stringhe presenti nello statement DATA ai nomi delle variabili citate nello statement READ. I nomi delle variabili nello statement READ ed i valori nello statement DATA sono separati da virgole. Di conseguenza si possono lasciare spazi aggiuntivi tra i valori poichè è la virgola che stabilisce la fine dei valori stessi.

READ A,B,C ricerca i primi tre valori in DATA. Se READ A,B,C viene eseguito di nuovo i successivi tre valori dello statement DATA vengono assegnati rispettivamente ad A,B,C e l'accoppiamento tra variabili e dati continua finchè non sono stati letti tutti i dati.

Formati : DATA costante_aritmetica, |costante_aritmetica|
DATA costante_di stringa, |costante_di stringa|

Esempi : 140 DATA 55, 793, 666, 947, 55
150 DATA CONTO, ETA', """NOME""", NUMERO DI CODICE FISCALE

Le costanti aritmetiche e le costanti di stringa nello statement DATA sono assegnate ai nomi di variabili dallo statement READ. La virgola viene usata per separare le introduzioni che si vogliono fare con READ/DATA. Si possono usare più statement DATA. Il primo dato di DATA viene assegnato al primo nome di variabile che si incontra in READ; il secondo dato di DATA al secondo nome di variabile, e così via.
Una volta letti tutti i dati, se il programma tenta di leggere altri dati che non esistono, si verifica un errore "di superamento in DATA". Per controllare questa condizione di errore, può essere usato lo statement ERR.
Se in una stringa di uno statement DATA è inserita una virgola, allora tutta la stringa deve essere racchiusa tra virgolette, altrimenti tale virgola potrebbe essere confusa con la virgola usata per separare i dati. Le virgolette non sono richieste se una stringa utilizza valori numerici come dati di stringa.

Esempio di READ/DATA:

```
100 FOR J = 1 TO 3
120 READ A$, A
130 PRINT A$, A
140 NEXT J
150 DATA FRED, 50, JACK, 20, JANE 200
900 PRINT "FINE DEI DATI"
910 END
```

☐ REM o ! o '

Formato : REM

Esempi :

```
10 REM QUESTO PROGRAMMA CALCOLA L'AREA DI UNA SFERA
20 LET R = 25! Stabilisci un valore iniziale
30 GOSUB 225'VAI ALLA ROUTINE DI CALCOLO
65 PRINT : REM STAMPA IL RAGGIO
```

Formato : ! e '

Esempi :

```
10 PRINT "ESEMPIO "!, COMMENTO DI CODA
20 GOTO 10! USA! e'
```

Il punto esclamativo (!) e l'accento (') vengono usati dopo uno statement per inserire commenti. REM deve iniziare a destra del numero di riga o di due punti, mentre ! e' non richiedono un segno di due punti precedenti.
REM, !, e ' vengono usati per fare commenti e note su un programma.
L'istruzione REM non viene eseguita dal programma.

Sebbene gli statement REM usino memoria aggiuntiva, sono un valido aiuto per leggere e documentare un programma.

☐ RESTORE

Formato : RESTORE |numero_di riga|

Esempi :

```
440 RESTORE 770
550 RESTORE
```

Lo statement RESTORE viene usato se si prevede che certi dati possano essere riutilizzati dal programma. Quindi, lo statement RESTORE permette di usare lo stesso statement DATA per più volte.
Senza lo statement RESTORE, se si tenta di leggere (READ) dati una seconda volta, si verifica un errore di "superamento in DATA". I dati possono essere ripristinati iniziando da un particolare numero di riga, attraverso l'opzione "numero_riga". Il seguente programma dirigerà l'esecuzione alla riga 50, quando incontra RESTORE 50.

```
10 REM  ESEMPIO DI READ - DATA - RESTORE
20 DIM  A (15)
30 FOR  I = 1 TO 10: READ A (I) : PRINT A (I): NEXT I
40 DATA 1, 2, 3, 4, 5
50 DATA 6, 7, 8, 9, 10
60 DATA 11, 12, 13, 14, 15
70 RESTORE 50
80 FOR N = 1 TO 10 : READ A (N) : PRINT A (N); : NEXT N
```

☐ RESUME

Formati : RESUME |numero di riga|
RESUME |NEXT|
RESUME

Esempi : 300 RESUME 55
440 RESUME NEXT
450 RESUME

RESUME è l'ultimo statement della routine di gestione degli errori ON ERROR riferiti al numero di riga. RESUME trasferisce il controllo al numero di riga indicato.

RESUME NEXT trasferisce l'esecuzione allo statement successivo a quello in cui si è verificato l'errore.

RESUME trasferisce l'esecuzione al numero di riga che ha causato l'errore, se non è stato specificato NEXT o un numero di riga dopo RESUME.

☐ RETURN

Formato : RETURN

Esempio : 550 RETURN

RETURN riporta il programma al numero di riga specificato dopo lo statement GOSUB, che aveva trasferito l'esecuzione a questo gruppo di statements.

Esempio di programma:

```
10 X = 1
20 GOSUB 80
30 PRINT X
40 X = 3
50 GOSUB 80
60 PRINT X
70 STOP
80 X = X *2
90 RETURN
```

☐ STACK

Formato : STACK

Esempi : 120 PRINT STACK! Stampa il numero di entrate disponibili nel contatore dei tempi
310 IF STACK = 0 THEN PRINT "CONTATORE PIENO"

La funzione STACK fornisce il numero di entrate ancora disponibili nel contatore. Il contatore dei tempi può contenere 20 impulsi da 1/60 ciascuno.

STACK viene usato per memorizzare gli intervalli di tempo, in sessantesimi di secondo, usati con SOUND e AFTER.

☐ STOP

Formato : STOP

Esempio : 190 STOP

STOP viene usato per arrestare l'esecuzione di un programma in un punto che non sia l'ultima riga del programma e visualizza il numero di riga a cui è stata interrotta l'esecuzione del programma. STOP è utile nella ricerca e nella correzione di errori perchè si può usare il comando PRINT in modo diretto per mostrare il valore delle variabili nel punto di arresto dell'esecuzione.

☐ VARPTR

Formati : VARPTR (nome di variabile)
VARPTR (PLM1)
VARPTR (PLM2)
VARPTR (CHR1)
VARPTR (CHR2)
VARPTR (RESERVE)

Esempi :
```
110 A = VARPTR (A$)
110 PRINT VARPTR (A$) + 1
120 J = VARPTR (TOTAL)
120 I = VARPTR (CHR2)
155 POKE VARPTR (RESERVE), & FE
```

Se l'argomento di questa funzione è un nome di variabile, la funzione restituisce il suo indirizzo nella tabella delle variabili. Quando la variabile è aritmetica, VARPTR restituisce l'indirizzo iniziale di 2 byte della variabile (byte più significativo e byte meno significativo). Quando la variabile è una stringa, VARPTR fornisce il numero di

byte della stringa, mentre la posizione iniziale della stringa viene fornita in VARPTR (A&) + 1(byte meno significativo) e VARPTR (A&) +2 (byte più significativo).

Bisogna notare che solo nel caso di stringhe l'indirizzo viene fornito nella notazione 6502 specificando il byte di bassa memoria prima del byte di alta memoria.
Eccetto che nel caso delle stringhe, l'intero indirizzo viene fornito da VARPTR nel formato byte alto / byte basso.

Con VARPTR possono essere usate le seguenti parole chiave :

VARPTR (PLMn) Fornisce l'indirizzo (MSB, LSB) del primo byte riservato per PLMn.

VARPTR (CHRn) Fornisce l'indirizzo (MSB, LSB) del primo byte riservato per CHRn.

VARPTR (RESERVE) Fornisce l'indirizzo (MSB, LSB) del primo byte riservato per i programmi in linguaggio assemblatore.

Per assegnare spazio bisogna usare OPTION PLM1, OPTION PLM2, OPTION CHR1, OPTION CHR2 e OPTION RESERVEn. Quando si è usato OPTION per riservare spazio, si può usare VARPTR per conoscere il byte di inizio di tale spazio.

☐ WAIT...AND

Formato : WAIT indirizzo AND byte di maschera, byte di confronto

Esempio: 330 || WAIT || &D40B, &FF, 110! WAIT FOR VBLANK

WAIT arresta il programma finchè non si verificano certe condizioni. L'esecuzione resta in attesa finchè il "byte di confronto", AND il byte di mascheramento, non è uguale all'indirizzo del byte contenuto in memmoria.
WAIT è ideale se si vuole arrestare l'esecuzione finchè non si verifica la condizione VBLANK (una dettagliata spiegazione di VBLANK è contenuta in "De Re ATARI"). VBLANK si verifica ogni sessantesimo di secondo ed è composto da un numero di righe sottostanti l'area visibile di scansione. Se l'esecuzione resta in attesa (WAIT) finchè non si verifica un VBLANK, lo scorrimento dello schermo viene interrotto a metà scansione e viene prodotto un segnale luminoso intermittente.
Questa tecnica viene sfruttata per dare movimento a figure come spiegato nell'Appendice C, "Set di caratteri alternativi".
Un esempio dello statement WAIT, usato per controllare la temporizzazione dello scorrimento verticale dello schermo è contenuto nell'Appendice A.

5 Funzioni del Programma

☐ FUNZIONI NUMERICHE

• ABS

Formato : ABS (espressione)

Esempio : ABS (-7)

ABS restituisce il valore assoluto di un numero. Dopo aver eseguito questa funzione, il valore del numero è sempre positivo. Se con ABS viene valutato il numero -7 (7 negativo) il risultato sarà 7 (7 positivo).

• ATN

Formato : ATN (espressione_aritmetica)

Esempio : ? ATN (.66) Stampa l'arcotangente di .66 espresso come .583373 radianti.

ATN restituisce la funzione arcotangente dell'espressione aritmetica.

• COS

Formato : COS (espressione_aritmetica)

Esempio : ? COS (.95) Stampa il coseno di .95 espresso come .581683 radianti.

COS restituisce il coseno trigonometrico dell'espressione aritmetica.

• EXP

Formato : EXP (espressione_aritmetica)

Esempio : ? EXP (3) Stampa 20.0855

EXP restituire il numero di Eulero (e) elevato alla potenza data dall'espressione aritmetica tra parentesi.

• INT

Formato : INT (espressione_aritmetica)

Esempio : ?INT (5.3) Visualizza 5 sullo schermo televisivo.
?INT (-7.6) Visualizza -8 sullo schermo televisivo.

INT restituisce un numero intero per un'espressione aritmetica.
INT arrotonda sempre all'intero più vicino.

● LOG

Formato : LOG (espressione_aritmetica)

Esempio : ?LOG (5) Stampa il logaritmo naturale 1.60944.

LOG restituisce il logaritmo naturale (LOG) dell'espressione aritmetica non negativa citata tra parentesi. LOG (0) da il messaggio : Errore di richiamo funzione,LOG (1) è 2.32396E-8.

● RND

Formato : RND
RND(0) come RND
RND (numero intero)

Esempi : ? RND Stampa 6 cifre casuali dopo il punto decimale
RND (37) Stampa un numero compreso tra 1 e 37

RND fornisce numeri casuali. RND e RND(0) forniscono numeri casuali compresi tra ma non comprendenti 0 e 1.
RND (numero intero) fornisce un numero intero positivo compreso tra 1 e il numero intero citato,estremi compresi.

● SGN

Formato : SGN (espressione_aritmetica)

Esempio : ? SGN (-34) Visualizza -1 sullo schermo televisivo

SGN restituisce il segno dell'espressione aritmetica racchiusa tra parentesi. Il segno è +1 se il numero tra parentesi è positivo, 0 se il numero è 0 e -1 se il numero è negativo.

● SIN

Formato : SIN (espressione_aritmetica)

Esempio : ? SIN (1) Stampa il seno di 1 espresso come 0.841471 radianti.

SIN restituisce il seno trigonometrico dell'espressione aritmetica.

● SQR

Formato : SQR (espressione_aritmetica)

Esempio : ? SQR (25) visualizza 5 sullo schermo televisivo.

SQR restituisce la radice quadrata di un'espressione aritmetica positiva racchiusa tra parentesi. Se l'espressione aritmetica valutata da SQR ha un segno negativo (-), si ottiene il messaggio: Errore di richiamo funzione.

● TAN

Formato : TAN (espressione_aritmetica)

Esempio : ? TAN (.22) Stampa la tangente di .22 espresso come .223619 radianti.

TAN restituisce la tangente trigonometrica dell'espressione aritmetica citata tra parentesi.

□ FUNZIONI DI STRINGA

+ (OPERATORE DI CONCATENAZIONE)

Formato : stringa + stringa

Esempio : 110 C$ = A$ + B$

Il simbolo più viene usato per unire due stringhe.

Programma di esempio :

```
110 A$ = "mai"
120 B$ = "più"
130 Z$ = A$ + B$
140 PRINT Z$

RUN RETURN
maipiù
```

● ASC

Formato : ASC (espressione_di stringa)

Esempio : ? ASC ("Smith") ! stampa 83 (codice decimale ATASCII per la lettera S)

ASC fornisce un codice ATASCII in decimali per il primo carattere della stringa. Vedere Appendice K per il set dei caratteri ATASCII.

● CHR$

Formato : CHR $ (numero_di codice _ ATASCII)

Esempio : 110 PRINT CHR$ (123) ! Stampa il simbolo ATASCII di picche.
100 PRINT CHR$ (65) ! Stampa il carattere A ATASCII

CHR$ converte un valore ATASCII in una stringa di carattere.
CHR$ è l'opposto della funzione ASC. Il "numero_di codice_ATASCII" può

essere qualsiasi numero complesso tra 0 e 255. L'Appendice K fornisce una tabella contenente il set di caratteri e i corrispondenti numeri di codice ATASCII.

- INKEY$

Formato : INKEY$

Esempio : 110 A$ = INKEY$

INKEY$ restituisce l'ultimo tasto premuto. Se non è stato premuto alcun tasto sulla tastiera, viene restituita una stringa vuota. Nel programma di esempio, lo statement 110 controlla se la stringa è vuota, attraverso un doppio segno di virgolette, senza spazi tra loro.

Il Microsoft BASIC II Atari non riconosce la barra spaziatrice, poichè INKEY$ elimina i blanks (spazi vuoti) iniziali e finali.

Programma di esempio :

```
100 A$ = INKEY$
110 IF A$ <> ""THEN PRINT" Hai battuto"; A$
120 GOTO 100
```

- INSTR

Formato : INSTR (m, A$, B$)

Esempio : 110 HOLD = INSTR (5, C$, B$)

INSTR ricerca una sottostringa B$ all'interno di una stringa più grande. La ricerca inizia all'm-esimo carattere. Se m non è specificato,la ricerca parte dal primo carattere della stringa. La funzione restituisce un numero che rappresenta la posizione del primo carattere di B$ incontrato all'interno di A$ o uno 0 se non esiste la sottostringa B$.

- LEFT$

Formato : LEFT$ (espressione_di_stringa_$, n)

Esempio : 100 A$ = "TOTALAMOUNT"
          110 PRINT LEFT$ (A$, 5)

LEFT$ restituisce gli n caratteri più a sinistra dell'espressione di stringa.

- LEN

Formato : LEN (espressione_di_stringa_$)

Esempio : 100 A$ = "COUNT THE"
          120 ? LEN (A$ + "CHARACTERS")! stampa il numero totale di
          130 ! caratteri ossia 20

LEN restituisce il numero totale di caratteri presenti nella stringa

indicata. LEN è l'abbreviazione di lunghezza. Vengono conteggiati spazi, numeri e simboli speciali. (In inglese length).

● MID$

Formato : MID$ (espressione_di_sringa_$, m,n)

Esempio : 100 A$ = "GETTHEMIDDLE"
          110 PRINT MID (A$, 4, 3)

MID$ estrae una porzione di una stringa. La stringa è identificata dal primo parametro della funzione. Il secondo parametro indica il carattere di inizio della stringa da estrarre. Il terzo parametro indica la sua lunghezza.

Programma di esempio :

```
110 A$ = "VALORE DELL'INTERÈSSE PAGATO"
120 B$ = MID$ (A$, 13, 9) ! VIENE STAMPATO : INTERESSE
130 PRINT B$
```

● RIGHT$

Formato : RIGHT$ (espressione_di_stringa_$, n)

Esempio : 100 A$ = "LA DESTRA"
          110 PRINT RIGHT$ (A$, 6)

RIGHT restituisce gli n caratteri più a destra di un'espressione di stringa.

● SCRN$

Formato : SCRN$ (X,Y)

Esempio : 10 ? SCRN$ (5,5)

Il carattere alla coordinata X e Y viene restituito come valore della funzione nel modo grafico per carattere. In altri modi grafici, SCRN$ dà il numero del registro del colore del "pixel" nella posizione X,Y.

Esempio di SCRN$ (X,Y) in un modo grafico di carattere.

```
10  GRAPHICS 1
20 COLOR 1
30 PRINT  # 6, AT (5,5); "A"
40 A$ = SCRN$ (5,5)
50 PRINT "Il carattere è: "; A$
60 END
```

Esempio di SCRN$ (X,Y) in modo grafico:

```
100 GRAPHICS 7
110 COLOR 3
120 PLOT 5,5
```

```
130 A$ = SCRN$ (5,5)
140 PRINT "IL REGISTRO DEL COLORE E'"; ASC (A$)
150 END
```

N O T A

Usare la funzione LEN per controllare che la stringa risultante non sia vuota (registro colore : ZERO).

- STR$

Formato : STR$ (espressione_aritmetica)

Esempio : 100 A = 999.02
110 PRINT STR$ (A)

STR$ converte un'espressione aritmetica in una stringa. Le operazioni di stringa possono poi essere eseguite con le stringhe risultanti. Va notato che quando due stringhe contigue vengono unite con il simbolo di concatenazione, rimane uno spazio tra di esse che rappresenta il segno del numero positivo.

Programma di esempio :

```
100 NUM1 = .22.344
120 NUM2 = 43.2
130 PRINT STR$ (NUM1) + STR$ (NUM2)
140 END

RUN RETURN
-22.344  43,2
```

- STRING$(n, A$)

(Disponibile solo con il dischetto di Estensione)

Formato : STRING$ (n, A$)

Esempio : 100 A$ = STRING$ (20,"*")

STRING$ (n, A$) fornisce una stringa composta da n ripetizioni di A$.

- STRING$(n, m)

(Disponibile solo con il dischetto di Estensione)

Formato : STRING$ (n, m)

Esempio : 110 PRINT STRING$ (20,123) ! Stampa 20 picche

STRING$ (n, m) fornisce una stringa composta da n ripetizioni di CHR$ (m).

- TIME$

Formato : TIME$ - "ore : minuti : secondi" tempo trascorso

Esempio : 100 PRINT TIME$

TIME$ inizializza il tempo nel formato "ore : minuti : secondi" e lo tiene costantemente aggiornato (± 90 sec. per 24 ore).

Esempi : 110 TIME$ = "22:55:05"
         120 TIME$ = "05:30:09"

N O T A

Usare degli zeri per rendere le ore, i minuti e i secondi numeri di 2 cifre.

Dopo aver inizializzato il tempo con TIME$, può essere usato nel programma.
TIME$ viene aggiornato in continuazione.Per esempio :

```
100 GRAPHICS 2
110 TIMES$=11:59:05"
120 PRINT#6, AT(3.3°"OROLOGIO DIGITALE"
130 PRINT#6, AT(,4)TIMES$
140 GOTO 120
```

● VAL

Formato : VAL (espressione_di stringa_numerica$)

Esempio : 100 B$ = "309"
          120 ? VAL (B$) ! stampa il numero 309
          130 END

VAL converte stringhe in valori numerici. VAL restituisce il valore numerico della costante numerica associata con "espressione_di stringa_numerica". Eventuali spazi iniziali e finali della stringa vengono ignorati. Le stringhe vengono convertite fino al primo carattere non numerico incontrato. Per esempio ; PRINT VAL ("123ABC") stampa 123. Se il primo carattere dell'espressione di stringa non è numerico, viene restituito il valore 0 (zero).

C] FUNZIONI SPECIALI

● EOF

Formato : EOF(n)
Esempio : 120 IF EOF(4) = THEN GOTO 60

Viene restituito il valore vero (1) o falso (0) per indicare che è stata rilevata la condizione di fine-file sull'ultima lettura dell'n-esimo IOCB.

● ERL

Formato : ERL
Esempio : 100 PRINT ERL

ERL fornisce il numero di riga dell'ultimo errore incontrato.

● ERR

Formato : ERR
Esempio : 120 PRINT ERR
          150 IF ERR 135 THEN GOTO 350

ERR fornisce il codice di errore dell'ultimo errore incontrato.

● FRE(Ø)

Formato : FRE(Ø)
Esempio : PRINT FRE(Ø)

Questa funzione fornisce il numero dei bytes di memoria che sono liberi e disponibili per l'utente. Viene usato principalmente in modo diretto con una variabile fittizia (0) per informare sulla quantità di spazio rimasto disponibile in memoria. Naturalmente FRE può essere anche usato all'interno di un programma BASIC, in modo differito.
L'uso di FRE(0) rilascia le posizioni di memoria di stringa che non vengono utilizzate. Questo utilizzo di FRE(0) per rilasciare le stringhe sparse viene chiamato "garbage collection (raccolta di rifiuti)".

● PEEK

Formato : PEEK(indirizzo)
Esempio : 110 PRINT PEEK(1034)
          135 PRINT PEEK(ADDR)

PEEK(&FFF) ricerca l'indirizzo racchiuso tra parentesi, in questo caso l'indirizzo FFF esadecimale. PEEK viene usato per esaminare il contenuto di una particolare posizione di memoria. Si può esaminare sia la memoria ROM che la memoria RAM. PEEK restituisce sempre un valore

Esempio :

PRINT PEEK (888)	Stampa il contenuto della posizione di memoria 888 (decimale)
PRINT PEEK (&FFFF)	Stampa il contenuto dell'indirizzo esadecimale FFFF.

● POKE

Formato : POKE indirizzo, byte

Esempi : POKE 2598, 255
110 POKE ADDR3, &FF
120 POKE PLACE,J

POKE scrive dati in un specificato indirizzo di memoria. L'indirizzo e il byte possono essere espressi come numeri decimale o esadecimali. Lo indirizzo e il byte possono anche essere espressioni. Perciò, se X*Y-2 rappresenta un indirizzo o un byte valido, può essere usato.

Esempi :

POKE &FFF, 43	Scrive il numero 43 all'indirizzo di memoria FFF (esadecimale)
X = 22	
Y = &&F	
POKE X, Y	Scrive il numero esadecimale 8F all'indirizzo di memoria 22 (decimale)

Va notato che la notazione decimale ed esadecimale sono solo due modi per assegnare un valore ad un byte di 8-bit. Il massimo numero ammesso per un byte è FF in esadecimale e 255 in decimale.

● STATUS

Formato : STATUS (numero_iocb)
STATUS ("unità:nome_programma")

Esempio : 100 A = STATUS(6)
120 A = STATUS("D:MICROBE.BAS")

STATUS restituisce il valore del quarto byte del blocco iocb (byte di stato). Il bit più significativo vale 1 per condizioni di errori; zero per condizioni non di errori. I rimanenti bits rappresentano un codice di errore :

TABELLA 5-1 ELENCO DI CODICI DI STATO

Esa	Dec	Significato
01	001	Operazione completata (nessun errore)
03	003	Fine-file (EOF)
80	128	Interruzione mediante tasto BREAK
81	129	IOCB già in uso (OPEN)
82	130	Unità non esistente
83	131	Aperto solo per scrittura
84	132	Comando non valido
85	133	Unità o file non aperto
86	134	Numero IOCB non valido (solo registro Y)
87	135	Aperto solo per lettura
88	136	Incontrato un fine file (EOF)
89	137	Record troncato
8A	138	Fine del tempo disponibile all'unità (non risponde)
8B	139	Unità NAK
8C	140	Errore di trasmissione dell'input nel bus seriale
8D	141	Cursore fuori dalle posizioni disponibili
8E	142	Errore di sovraccarico dei dati trasmessi sul bus seriale
8F	143	Errore nel totale di controllo dei dati trasmessi sul bus seriale
90	144	Errore generato dall'unità
91	145	Modo video errato
92	146	Funzione non supportata dal gestore
93	147	Memoria insufficiente per il modo "video"
A0	160	Numero dell'unità a dischi errato
A1	161	Troppi file aperti sul disco
A2	162	Disco pieno
A3	163	Errore fatale di I/O su disco
A4	164	Mancata corrispondenza del numero di file interno
A5	165	Errore nel nome del file
A6	166	Errore della lunghezza dei dati
A7	167	File bloccato
A8	168	Comando non valido per dischi
A9	169	Directory piena (64 files)
AA	170	File non trovato
AB	171	Puntatore non valido

- TIME

Formato : TIME

Esempio : 200 PRINT TIME

TIME fornisce il contenuto delle posizioni riservate all'orologio di sistema (RTCLOCK). Le posizioni decimali 18, 19 e 20 (RTCLOCK) mantengono il tempo del sistema in sessantesimi di secondo. TIME restituisce

6 cifre decimai. La differenza tra TIME$ e TIME è la seguente :

TIME fornisce il tempo standard in ore, minuti e secondi.

TIME dà il tempo in sessantesimi di secondo.

- USR

Formato : USR (indirizzo, nl)

Esempio : 550A = USR (898, Ø)

La funzione USR permette di trasferire l'esecuzione del programma ad una routine in linguaggio macchina. Si tratta di una funzione di programmazione avanzata che permette di utilizzare in modo vantaggioso tutte le funzioni speciali del calcolatore.

La funzione USR ha due parametri : il primo è un indirizzo di memoria e il secondo è un valore opzionale, nl. Il valore di nl è normalmente l'indirizzo di una tabella dati, ma può anche essere un valore passato alla routine per svolgere una determinata azione. Una volta eseguita la funzione USR, i parametri vengono memorizzati in &E3 e &E4. Il seguente programma cambia colore al video ed è eseguito alla velocità tipica del linguaggio macchina.

Programma di esempio :

```
10!  ROUTINE DI CONTROLLO CHIAMATA ALLA FUNZIONE USR
20!  ROUTINE ASSEMBLER MEMORIZZATA IN MEMORIA
30!  LA ROUTINE E' :
40!  LDA # 35
50!  STA 710
60!  RTS
70!
80!
90!
100  A = Ø:I = Ø:COL = Ø:C = Ø
110  OPTION RESERVE 10
120  ADDR = VARPTR(RESERVE) !INDIRIZZO INIZIALE
130  FOR I = 0 TO 5
140  READ A
150  POKE ADDR + I,A
160  NEXT I
170  DATA &A9,&23,&8D,&C6,&02,&60
180  A = USR(ADDR,VARPTR(I))
190  STOP
```

6 Funzioni dei Giochi

☐ GENERALITA'

Il comando GRAPHICS seleziona uno dei 12 modi grafici disponibili, numerati da 0 a 11 con il chip GITIA e da 0 a 8 con il chip CTIA. Una descrizione dettagliata di GTIA e CTIA è contenuta nel Manuale "De Re ATARI".

L'espressione aritmetica che segue GRAPHIC deve essere valutata come un numero intero positivo. Il modo grafico 0 è un modo a schermo completo. Lo standard per il Microsoft BASIC II Atari, è GRAPHIC 0.

I comandi GRAPHICS da 1 fino a 8 sono modi grafici a schermo suddiviso nella parte inferiore dello schermo c'è un'area di testo di 4 righe.

GRAPHICS 0, GRAPHICS 1, GRAPHICS 2 visualizzano caratteri aventi diverse dimensioni. GRAPHICS 0 visualizza caratteri di dimensione regolare. GRAPHICS 1 visualizza caratteri di doppia larghezza. GRAPHICS 2 visualizza caratteri a doppia larghezza e altezza.
Caratteri grafici (caratteri ottenuti con il tasto CONTROL) non possono essere visualizzati in GRAPHICS 1 o 2 a meno che non si modifichi l'indirizzo base dei caratteri (POKE 756, 226).

I comandi da GRAPHICS 3 fino a GRAPHICS 11 sono modi per tracciare punti direttamente sullo schermo televisivo. Il modo grafico determina la dimensione dei punti e il numero di colori del campo di gioco che si vogliono usare. Il numero massimo di colori del campo di gioco, nei modi a tracciatura di punti, è quattro. Ma è possibile ottenere altri quattro colori sullo schermo usando giocatori e missili. Per informazioni sulla grafica missile e giocatore, far riferimento al Capitolo 7.

I comandi da GRAPHICS 9 fino a 11 sono disponibili solo se il sistema ha un chip GTIA. GRAPHICS 9 permette di avere un solo colore con 16 luminanze. Con GRAPHICS 10 si possono ottenere nove colori per campo di gioco con 8 luminanze . Con GRAPHICS 11 si possono ottenere 16 colori con una sola luminanza.

☐ GRAPHICS

Formato : GRAPHICS espressione_aritmetica

```
Esempi : GRAPHICS 2
         100 GRAPHICS 5 + 16
         170 GRAPHICS 1 + 32 + 16
         120 GRAPHICS 8
         150 GRAPHICS 0
         140 GRAPHICS 18
```

GRAPHICS va usato per selezionare uno dei modi grafici (da 0 a 11). La Tabella 6-1 riassume i dodici modi e le principali caratteristiche di ognuno di essi.

GRAPHICS 0 è un modo per visualizzare testi a schermo completo, come pure stampare caratteri usando lo statement PRINT. I modi da GRAPHICS 1 a GRAPHICS 8 sono a schermo suddiviso. Tali modi attualmente includono quattro righe di GRAPHICS 0 nella parte inferiore dello schermo televisivo. Quest'area di testo utilizza lo statement PRINT. Per visualizzare dati in un'area grafica più grande, nei modi GRAPHICS 1 e GRAPHICS 2, usare PRINT # 6. Il seguente programma viene visualizzato nell'area grafica, nei modi GRAPHICS 1 o GRAPHICS 2:

```
100 GRAPHICS 1
110 PRINT#6, AT(4,4);"AREA GRAFICA"
120 PRINT "AREA DI TESTO"
```

Aggiungendo +16 ai modi da GRAPHICS 1 fino a GRAPHICS 11 si ottiene un modo grafico a schermo intero. Se si esegue il seguente programma senza la riga 140, lo schermo ritorna al modo grafico 0. Premere il tasto BREAK per uscire dal loop alla riga 140.

```
110 GRAPHICS 2+16
120 PRINT#6, AT(3,3);"L'INTERO VIDEO E'"
130 PRINT#6, AT(4,4);"GRAPHICS 2"
140 GOTO 140
BREAK
```

Normalmente lo schermo viene ripulito da tutti i precedenti caratteri grafici quando si incontra uno statement GRAPHICS n. Aggiungendo +32 si impedisce al comando di ripulire lo schermo.

I modi grafici da 3 a 11 sono modi grafici a tracciatura di punti che utilizzano gli statement COLOR n e PLOT.

Usando lo statement SET COLOR si possono cambiare i colori standard con qualsiasi delle 128 diverse combinazioni di colore/luminosità. I modi a tracciatura di punti sono illustrati in un esempio alla fine di questo capitolo.
Per tornare a GRAPHICS 0 in modo diretto, occorre battere GRAPHICS 0 e premere RETURN.

Tabella 6-1 Modi Grafici e Formati di schermo

	Modo Grafico	Tipo	Colonna	Righe** schermo suddiviso	Righe** schermo completo	Numero di colori	RAM richiesta (Byte)
CTIA	0	TESTO	40	-	24	1-1/2	992
	1	TESTO	20	20	24	5	674
	2	TESTO	20	10	12	5	424
	3	GRAFIC	40	20	24	4	434
	4	GRAFIC	80	40	48	2	694
	5	GRAFIC	80	40	48	4	1174
	6	GRAFIC	160	80	96	2	2174
	7	GRAFIC	160	80	96	4	4198
	8	GRAFIC	320	160	192	1-1/2	8112
GTIA	9	GRAFIC	80	-	192	1	8112
	10	GRAFIC	80	-	192	9	8112
	11	GRAFIC	80	-	192	16	8112

I comandi GRAPHICS da 3 a 11 tracciano singoli punti sullo schermo televisivo. Il numero che segue GRAPHICS stabilisce la dimensione dei punti da tracciare. GRAPHICS 3 ha i punti più grandi. Il programma di esempio può essere usato per visualizzare la dimensione dei punti nei modi 3-8.

Programma di esempio :

```
10 INPUT "QUALE MODO GRAFICO (3-8) ?";4
20 GRAPHICS G + 16
30 COLOR 1
40 PLOT 5, 5
45 FOR H = 1 TO 1900 : NEXT
50 GOTO 10
```

Se si inserisce un nuovo statement (statement 15), 15 SETCOLOR 4, 4,8, si otterranno grossi punti rosa al posto dei punti standard arancioni. Questa modifica al programma originale dà dei punti rosa perchè SETCOLOR 4, x, x si allinea con COLOR 1 in GRAPHICS 3.

☐ COLOR

Formato : COLOR n

Esempio : 100 COLOR 4

COLOR viene usato con PLOT per visualizzare fino a 4 colori sullo schermo televisivo. Per tracciare un colore bisogna codificare uno statement COLOR nei modi GRAPHICS da 3 a 11.

Quando si usa lo statement COLOR non preceduto da un comando SETCOLOR, si ottengono i colori standard (quelli che si trovano attualmente nei registri del colore). I registri dei colori vengono inizializzati sulla base della Tabella 6-2. Per esempio i colori standard per GRAPHICS 3 sono : arancione per il registro di colore 4, verde chiaro per il registro di colore 5, blu scuro per il registro 6 e nero per il registro 8.

NOTA

Bisogna sempre codificare uno statement COLOR per disegnare un punto nel campo da gioco, mentre SETCOLOR è necessario solo per ottenere un colore diverso da quello standard.

Tabella 6-2 Colori standard, modi, SETCOLOR e COLOR

Colori standard	Modo	Registro di colore	Nr. Colore	Descrizione e commenti
	GRAPHICS 0	4	Il registro	
Blu chiaro		5	mantiene	Luminanza carattere
Blu scuro		6	i caratteri	(la stessa dello sfondo)
		7		Carattere
Nero	Modo Testo	8	Bordo	
Arancione		4		Carattere
Verde chiaro	GRAPHICS 1,2	5		Carattere
Blu scuro		6		Carattere
Rosso		7		Carattere
Nero	Modo Testo	8		Sfondo, bordo
Arancione		4	1	Punto grafico
Verde chiaro	GRAPHICS 3,5,7	5	2	Punto grafico
Blu scuro		6	3	Punto grafico
		7		——
Nero	Modi a 4 colori	8	0	Sfondo, bordo
Arancione	GRAPHICS 4 e 6	4	1	Punto grafico
		5		——
		6		——
		7		——
Nero	Modi a 2 colori	8	0	Sfondo, bordo
	GRAPHICS 8	4		——
Blu chiaro		5	1	——
Blu scuro		6	0	——
		7		——
Nero	1 col/2 lumin	8		Bordo

Colori standard	Modo	Registro di colore	Nr. Colore	Descrizione e commenti
Nero	GRAPHICS 9	8	0-15	Punto grafico. Il valore del colore stabilisce la luminescenza
Nero	GRAPHICS 10	0	0	Punto grafico
Nero		1	1	Punto grafico
Nero		2	2	Punto grafico
Nero		3	3	Punto grafico
Arancione		4	4	Punto grafico
Verde chiaro		5	5	Punto grafico
Blu scuro		6	6	Punto grafico
Rosso		7	7	Punto grafico
Nero		8	8	Sfondo
Grigio	GRAPHICS 11	8	0-15	Il valore del punto grafico determina la gradazione

NOTA

Il colore dei grafici per il giocatore e missile è SETCOLOR registro, colore, luminanza, dove i valori di registro = 0, 1, 2, 3 impostano il colore del giocatore e missile rispettivamente a 0, 1, 2, 3. I grafici giocatori e missili funzionano in tutti i modi grafici.

☐ SETCOLOR

Formato : SETCOLOR registro,colore,luminanza

Esempio :330 SETCOLOR 5,4,10

Lo statement SETCOLOR associa un colore e una luminanza ad un registro di colore.
I registri di colore 0, 1, 2, 3 sono rispettivamente per i giocatori e missili 0, 1, 2, 3. I registri di colore 4, 5, 6, 7 assegnano colori al campo da gioco. Il registro 8 è sempre un registro di sfondo.
Il numero del colore può essere un numero qualsiasi compreso tra 0 e 15 (vedi tabella 6-3).
La luminanza del colore deve essere un numero pari compreso tra 0 e 14; più alto è il numero più brillante è lo schermo; 14 è quasi totalmente bianco.

Tabella 6-3 NUMERI E COLORI ATARI (SETCOLOR COMMAND)

Colori	SETCOLOR colore Numero (Decimale)	SETCOLOR colore Numero (Esadecimale)
Grigio	0	0
Arancio chiaro (oro)	1	1
Arancione	2	2
Rosso - arancio	3	3
Rosa	4	4
Viola	5	5
Viola -blu	6	6
Azzurro -blu	7	7
Blu cielo	8	8
Blu chiaro	9	9
Turchese	10	A
Verde acqua	11	B
Verde	12	C
Giallo - verde	13	D
Arancio - verde	14	E
Arancio chiaro	15	F

[] PLOT / PLOT...TO

Formato : PLOT X,Y
PLOT X,Y TO X,Y

Esempio : 100 PLOT 12,9
112 PLOT 6,9 TO 3,3

PLOT viene usato per eseguire disegni a punto singolo, disegnare righe e delineare oggetti sullo schermo televisivo.

PLOT utilizza un sistema di coordinate X-Y per specificare la posizione dei punti. La coordinata X rappresenta la linea orizzontale, mentre la coordinata Y rappresenta la colonna verticale (Vedere Tabella 6-1). Occorre fornire un numero il cui valore dia 0 al numero più grande associato al modo. Prima si definisce X e poi Y.

0,0 ——— ► X
↓
Y

L'istruzione PLOT può essere concatenata. Cioè un punto PLOT può essere usato per tracciare il successivo punto.
Il risultato sarà la concatenazione di due punti PLOT in una riga diritta. Usando punti concatenati diviene anche molto facile tracciare il contorno di un oggetto. Per concatenare punti, usare la pa

rola 10 tra due statement PLOT, X, Y.

Programma di esempio :

```
 90 COLOR 1              ! Usare un'istruzione
                         COLOR prima di PLOT
100 PLOT 5.5 TO 5.15     ! Disegnare una riga di-
                         ritta
```

Il seguente programma di esempio mostra il funzionamento degli statements PLOT, COLOR e SETCOLOR:

```
100 GRAPHICS 3 + 16 !IL 16 TOGLIE L'AREA DI TESTO
110 SETCOLOR 5, 4, 8 !ROSA
120 SETCOLOR 6, 0, 4 !GRIGIO
130 SETCOLOR 8, 8, 6 !BLU
140 COLOR 1 !COLORE 1 E' ARANCIONE (STANDARD)
150 PLOT 5,5 TO 10,5 TO 10,10 TO 5,10 TO 5,5 !ARANCIONE
160 COLOR 2 !ROSA
170 PLOT 7,7 TO 12,12 TO 2,12 TO 7,7
180 COLOR 3 !GRIGIO
190 PLOT 2,7 TO 12,7
200 GOTO 200
```

☐ FILL

Formato : FILL X, Y TO X, Y

Esempio : 550 FILL 10,10 TO 5,5

FILL riempie un'area con il colore indicato dagli statements COLOR e SETCOLOR. Il processo FILL esegue una spazzolata attraverso lo schermo televisivo da sinistra a destra. FILL arresta la colorazione dello schermo ed inizia una successiva spazzolata quando incontra una riga o un punto PLOT. La riga a sinistra dell'oggetto colorato viene specificata dallo schermo statement FILL.

Il seguente esempio illustra il funzionamento di FILL. Inizialmente vengono specificati i tre lati di una casella. PLOT 5,5 TO 20,5 TO 20, 20 TO 5,5 realizzano il lato superiore,quello in basso e quello a destra della casella. Il lato sinistro viene realizzato con lo statement FILL 5,5 TO 5,20.

```
5,5   ----------  20,5
      ^        |
      |        |
5,20  ----------  20,20
```

```
10 GRAPHICS 5
20 SETCOLOR 4,12,8 (Registro 4, verde, luminosità media)
30 COLOR 1 !COLOR 1 e accoppiato con SETCOLOR 4 in GRAPHICS 5
40 PLOT 5,5 TO 20,5 TO 20,20 TO 5,20
50 FILL 5,5 TO 5,20
60 END
```

La riga 40 nel precedente esempio esegue tre lati della casella.Poi lo statement FILL, riga 50, esegue il lato sinistro e riempie la casella. Il processo FILL esegue una scansione dalla riga indicata verso destra, finchè non raggiunge le righe PLOT.

[ ] CLS

Formato : CLS opzione _ registro _ di sfondo

Esempi : CLS
110 CLS
100 GRAPHICS 3, CLS & C5
330 CLS 25

CLS ripulisce le aree di testo dello schermo e posiziona il registro del colore di sfondo al valore indicato se specificato.

Nei modi GRAPHICS 0 e GRAPHICS 8 il numero opzionale specificato dopo CLS stabilisce il colore e la luminosità del bordo. In GRAPHICS 1,2,3, 4,5,6,7 il numero opzionale che segue CLS stabilisce il colore e la luminosità dello sfondo.

Tabella 6-4 CARATTERI NEI MODI GRAFICI 1 E 2

POKE 756,224	POKE 756,226	SETCOLOR 4	SETCOLOR 5	SETCOLOR 6	SETCOLOR 7
SPAZIO		32	0	160	128
!		33	1	161	129
"		34	2	162	130
#		35	3	163	131
$		36	4	164	132
%		37	5	165	133
&		38	6	166	134
'		39	7	167	135
(		40	8	168	136
)		41	9	169	137
*		42	10	170	138
+		43	11	171	139
,		44	12	172	140
-		45	13	173	141
.		46	14	174	142
/		47	15	175	143
0		48	16	176	144
1		49	17	177	145
2		50	18	178	146
3		51	19	179	147
4		52	20	180	148
5		53	21	181	149
6		54	22	182	150
7		55	23	183	151
8		56	24	184	152
9		57	25	185	153
:		58	26	186	154
;		59	27	187	155

Tabella 6-4 CARATTERI NEI MODI GRAFICI 1 E 2

POKE 756,224	POKE 756,226	SETCOLOR 4	SETCOLOR 5	SETCOLOR 6	SETCOLOR 7
<	↑	60	28	188	156
=	↓	61	29	189	167
>	←	62	30	190	168
?	→	63	31	191	169
@	◆	64	96	192	224
A	a	65	97	193	225
B	b	66	98	194	226
C	c	67	99	195	227
D	d	68	100	196	228
E	e	69	101	197	229
F	f	70	102	198	230
G	g	71	103	199	231
H	h	72	104	200	232
I	i	73	105	201	233
J	j	74	106	202	234
K	k	75	107	203	235
L	l	76	108	204	236
M	m	77	109	205	237
N	n	78	110	206	238
O	o	79	111	207	239
P	p	80	112	208	240
Q	q	81	113	209	241
R	r	82	114	210	242
S	s	83	115	211	243
T	t	84	116	212	244
U	u	85	117	213	245
V	v	86	118	214	246

Tabella 6-4 CARATTERI NEI MODI GRAFICI 1 E 2

POKE 756.224	POKE 756.226	SETCOLOR 4	SETCOLOR 5	SETCOLOR 6	SETCOLOR 7
W	w	87	119	215	247
X	x	88	120	216	248
Y	y	89	121	217	249
Z	z	90	122	218	250
[		91	123	219	251
\	|	92	124	220	252
]		93	125	221	253
^		94	126	222	254
_		95	127	223	255

Programmi di Esempio :

I seguenti programmi funzionano nei modi GRAPHICS 1 o GRAPHICS 2. I programmi mostrano l'insieme dei caratteri alternativi ed un insieme di caratteri speciali (POKE 756.226). Per rieseguire questi due programmi, premere il tasto BREAK e battere RUN, seguito da RETURN :

```
2 REM TASTIERA MACCHINA DA SCRIVERE
10 GRAPHICS 2
20 SETCOLOR 4,0,0! Per evitare il video pieno di cuori
30 PRINT "Batti : Verde/Blu/Rosso/(V/B/R)"
40 INPUT "E BATTI RETURN"; C$
50 IF C$ = "V" THEN K = 32
60 IF C$ = "B" THEN K = 128
70 IF C$ = "R" THEN K = 160
80 PRINT "BATTI MAIUSCOLO / MINUSCOLO(1/2)"
90 INPUT "E PREMI RETURN?"; B$
100 IF B$ = "1" THEN 120
110 POKE 756,226
120 PRINT "ORA BATTI - TASTI ALFABETICI + TASTI DI CONTROLLO (CTRL)
130 A$ = INKEY$
140 IF A$ = ""THEN 130
150 A = ASC(A$) + K!32 è Verde 128 è Blu, 160 è Rosso
160 PRINT A
170 PRINT #6, CHR$(A);
180 GOTO 130
```

```
100 REM LAMPO
110 GRAPHICS 16 + 2
120 X = RND (36)
130 ON ERROR GOTO 150
140 PRINT  #6, TAB (X); "*"
146 GOTO 120
150 GRAPHICS 32 + 16 + 2
160 RESUME
```

Il breve programma che segue illustra l'uso della Tabella 6-4. Questo programma stampa il codice ATASCII di un carattere nell'area di testo ed il carattere stesso nell'area grafica. Ogni volta che si preme il tasto RETURN, appare un nuovo carattere. La ragione per cui SETCOLOR 4,Ø,Ø e uguale a SETCOLOR 8,Ø,Ø è per evitare uno schermo pieno di cuoricini. Un altro modo è quello di abbassare il set di caratteri in RAM (usando il comando MOVE) e ridefinire il carattere di cuore come 8 per 8 zeri. Consultare l'Appendice C "Insiemi di caratteri alternativi" per vedere un esempio di come abbassare o ridefinire un insieme di caratteri.

Il set dei caratteri speciali è illustrato nel seguente programma. Per vedere il set dei caratteri standard, basta concellare la riga 20. L'istruzione GRAPHICS 2 trasferisce automaticamente il valore 224 nella locazione 756.

```
10 GRAPHICS 2
20 POKE 756.226
30 SETCOLOR 8,0,0
40 SETCOLOR 4,0,0 !EVITARE I CUORI A VIDEO
50 SETCOLOR 5,4,6 !ROSA
60 SETCOLOR 6,12,2 !VERDE + AREA TESTO
70 SETCOLOR 7,9,6 !BLU CHIARO
80 A$ - INKEY$
90 IF A$ = ""THEN 80
100 ON ERROR GOTO 150
110 PRINT #6, AT(6,6);CHR$(X)
120 PRINT X
130 X = X + 1
140 GOTO 80
150 RUN !RIPETE QUANDO  ARRIVA A 256
```

□ IL COMANDO SOUND

Formato : SOUND voce, freouenza, distorsione, volume, durata.

Esempi : 120 SOUND 2,204,10,12,244
100 SOUND 0,122,8,10

La voce può essere un numero da 0 a 3, cioè con il comando SOUND si possono usare fino a quattro voci.

La frequenza è un numero qualsiasi compreso tra 0 e 255 (Vedere Tabella 6-5).
La distorsione è un numero qualsiasi compreso tra 0 e 14. Lo standard è un tono puro. Per creare un tono "puro" viene usato un 10, mentre un dodici fornisce un suono simile ad un cicalino.

Il volume è un numero qualsiasi compreso tra 0 e 15. Si usa un 1 per creare un suono che sia appena udibile e un 15 per un suono forte. Un valore 8 fornisce un suono di intensità normale. Se si usano più statement SOUND, il volume totale non dovrebbe superare il 32. In caso contrario si avrebbe uno spiacevole suono monco.

La durata di un tono o di un rumore viene fornita in sessantesimi di secondo. Se la durata non viene specificata, il tono continua finchè il programma non raggiunge uno statement END un altro statement RUN, oppure finchè non viene introdotto un secondo statement SOUND che usi lo stesso numero di voce seguito da 0,0,0. Se uno statement INPUT segue uno statement SOUND, il suono continua finchè non viene eseguito lo statement INPUT. Un suono può anche essere arrestato premendo il tasto BREAK.

Esempio : SOUND 2,204,10,12
SOUND 2,0,0,0

Tabella 6-5 Frequenze di valori di tono

Note	Es.	Decimale	
NOTE ALTE	C	1D	29
	B	1F	31
	A# or B	21	33
	A	23	35
	G# or A	25	37
	G	28	40
	F# or G	2A	42
	F	2D	45
	E	2F	47
	D# or E	32	50
	D	35	53
	C# or D	39	57
	C	3C	60
	B	40	64
	A# or B	44	68
	A	48	72
	G# or A	4C	76
	G	51	81

Tabella 6-5 Frequenze dei valori di tono

Note	Es.	Decimale	
NOTE ALTE	F # or G	55	85
	F	5B	91
	E	60	96
	D# or E	66	102
	D	6C	108
	C # or D	72	104
MEDIE C	C	79	121
	B	80	128
	A # or B	88	136
	A	90	144
	G # or A	99	153
	G	A2	162
	F # or G	AD	173
	F	B6	182
NOTE BASSE	E	C1	193
	D # or E	CC	204
	D	D9	217
	D # or D	E6	230
	C	F3	243

Programma di esempio :

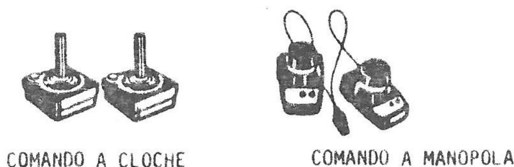
```
10 GRAPHICS 2+16
20 SETCOLOR 4,8,4
30 PRINT#6, AT(3,3);
40 FOR DELAY = 1 TO 1000:NEXT
50 GRAPHICS 2+16
60 PRINT#6, AT(3,3);"AT THE CAPE"
70 FOR DELAY = 1 TO 1000:NEXT
80 GRAPHICS 0
90 POKE 752.1
100 SETCOLOR 6,0,0
110 FOR T = 1 TO 24:PRINT "":NEXT
120 PRINT TAB(11);CHR$(8);CHR$(10)
130 PRINT TAB(11);CHR$(22);CHR$(2)
140 PRINT TAB(11);CHR$(22);CHR$(2)
150 PRINT TAB(11);CHR$(13);CHR$(13)
160 PRINT TAB(11);CHR$(6);CHR$(7)
170 FOR VOL = 15 TO 0 STEP -1
180 SOUND 2,77,8,VOL
190 PRINT CHR$(155)!ALZA IL RAZZO
200 FOR R = 1 TO 200:NEXT R
```

```
210 NEXT VOL
220 END
```

Il precedente programma è un esempio di uso dello statement SOUND.Esso diminuisce (per mezzo di un loop) il volume di un suono distorto. L'effetto del suono somiglia a quello di un razzo che parte per lo spazio.

□ COMANDI PER I GIOCHI

Nel Microsoft BASIC II Atari l'istruzione PEEK legge i comandi per i giochi. I comandi sono collegati direttamente ai jack di governo del'Home Computer Atari. Agli indirizzi di PEEK possono essere dati gli stessi nomi elencati di seguito oppure brevi nomi di variabile. Un elenco completo degli indirizzi PEEK viene fornito nell'Appendice E. Per definire i comandi dei comandi a manopola e a cloche si può anche usare il comando DEF (consultare DEF nel capitolo 4, per informazioni sulle funzioni definite dall'utente).



COMANDO A CLOCHE   COMANDO A MANOPOLA

Fig.6-1 Comandi giochi

● COMANDI A MANOPOLA

Il seguente programma di esempio legge e stampa lo stato del comando a manopola Ø (il primo nel bocchettone più a sinistra). Questo esame della memoria (PEEK) può essere usato con altre funzioni o comandi per generare altre azioni come ad esempio: suono, grafica, ecc. Un esempio può essere IF PADDLE (Ø) > 14 THEN GOTO 440. Ricercando l'indirizzo del comando a manopola con il comando PEEK, viene restituito un valore compreso tra 1 e 228, e tale numero aumenta man mano che la manopola del comando viene ruotata in senso antiorario.
Esempio di inizializzazione e di uso dello statement PEEK per PADDLE (Ø):

```
10 PADDLE(Ø) = 624
20 PRINT PEEK (PADDLE(Ø))
30 GOTO 2Ø
```

Numero PADDLE (manopola) e indirizzi PEEK (decimali)

PADDLE (Ø) = 624
PADDLE (1) = 625
PADDLE (2) = 626
PADDLE (3) = 627
PADDLE (4) = 628
PADDLE (5) = 629
PADDLE (6) = 630
PADDLE (7) = 631

La lettura dei seguenti indirizzi restituisce uno stato Ø se si preme il pulsante a scatto (grilletto) del comando selezionato. Altrimenti, restituisce il valore 1.
Esempio dell'uso del grilletto del comando a manopola (Ø):

```
10 PTRIG (Ø) &27C
20 PRINT PEEK (PTRIG(Ø))
30 GOTO 2Ø
```

Numero di PTRIG (scatto grilletto) e indirizzi PEEK (decimali)
PTRIG (Ø) = 636
PTRIG (1) = 637
PTRIG (2) = 638
PTRIG (3) = 639
PTRIG (4) = 64Ø
PTRIG (5) = 641
PTRIG (6) = 642
PTRIG (7) = 643

- COMANDI A CLOCHE

Lo statement PEEK per la ricerca degli indirizzi dei governi a cloche opera come sui governi a manopola. I governi a cloche sono numerati da 0 a 3 da sinistra a destra.
Esempio dell'uso del governo a cloche (Ø) :

```
10 STICK (Ø) = 632
20 PRINT PEEK (STICK (Ø))
30 GOTO 2Ø
```

Numero di STICK (cloche) e indirizzi PEEK (decimali)
STICK (Ø) = 632
STICK (1) = 633
STICK (2) = 634
STICK (3) = 635

La Figure 6-2 indica il numero di PEEK che viene restituito per le varie posizioni della cloche :

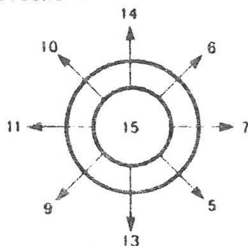


Fig. 6-2 Valori di scatto del comando a cloche

Lo scatto della cloche funziona come il pulsante a scatti della manopola.

Uso dello scatto della cloche (Ø) :

```
10 STRIG (Ø) = 64
20 PRINT PEEK (STRIG(Ø))
30 GOTO 2Ø
```

Numero di STRIG (scatto grilletto) e indirizzi PEEK (decimali)

STRIG (Ø) = 644
STRIG (1) = 645
STRIG (2) = 646
STRIG (3) = 647

```
10 REM QUESTO PROGRAMMA FARA' "BANG!"
15 REM QUANDO IL BOTTONE ROSSO DEL GOVERNO A CLOCHE VIENE PREMUTO
20 IF PEEK (644) = Ø THEN ? "BANG! "
30 IF PEEK (644) = 1 THEN  CLS
40 GOTO 20
```

● TASTI FUNZIONE SPECIALI

Il seguente programma usa i tasti montati sul lato destro del Calcolatore ATARI:

```
10 POKE 53279,Ø
20 PRINT PEEK (53279)
30 GOTO 2Ø
```

Il comando PEEK all'indirizzo 53279 (decimale) restituisce un numero che indica quale tasto è stato premuto :

7 = Nessun tasto
6 = Tasto START
5 = Tasto SELECT
3 = Tasto OPTION

7 Introduzione alla Grafica Giocatore e Missile

L'Home Computer Atari ha funzioni speciali incorporate che interessano funzioni di grafica e di animazione. Queste funzioni vengono normalmente chiamate "grafica giocatore e missile". I termini "giocatore" e "missile" sono derivati dalle funzioni di animazione presenti nei videogiochi ATARI. Le tabelle binarie del giocatore e missile risiedono nella memoria RAM destinata a queste funzioni. Tale RAM contiene quattro giocatore da 8-bit e quattro missili da 2-bit (vedere Fig.7-1). Ad ogni missile è associato un giocatore, a meno che non si scelga di unire tutti i missili per formare un quinto giocatore indipendente (vedere "Controllo di priorità").

Un giocatore, come la nave spaziale illustrata nella Fig. 7-2, viene visualizzato mappando la sua tabella binaria direttamente sullo schermo, sul campo di gioco. Il primo byte della tabella viene tracciato sulla riga superiore dello schermo, il secondo byte sulla seconda riga e così via.
Ogni volta che sulla tabella appaiono degli 1, vengono attivati i "pixels" dello schermo; quando invece appaiono degli 0, i "pixels" rimangono spenti. Il susseguirsi di "pixels" chiari e scuri crea l'immagine.

La grafica giocatore e missile può essere visualizzata con risoluzioni a riga singola (usando OPTION(PLM1)) o a riga doppia (usando OPTION (PLM2)). Se si sceglie la risoluzione a riga singola, ogni byte del giocatore viene visualizzato su una singola riga di scansione. Se si sceglie la risoluzione a doppia riga, ogni byte occupa due righe ed il giocatore appare più grande che nella risoluzione a riga singola.
Ogni giocatore è lungo 256 bytes con la risoluzione a riga singola o 128 bytes con la risoluzione a doppia riga. Il tipo di risoluzione deve essere programmato una sola volta e, una volta selezionata, viene applicato a tutti i giocatore e missile del programma.
Il Programma Dimostrativo della Grafica giocatore e missile compreso in questa sezione è un esempio di programmazione con risoluzione a doppia riga.

La grafica giocatore e missile fornisce una notevole flessibilità nella programmazione di grafici in movimento su video. Vengono forniti dei registri per il colore, per il dimensionamento del giocatore e del missile, per il posizionamento orizzontale, per la priorità giocatore-campo di gioco e per il controllo della collisione.

I seguenti comandi BASIC II servono per costruire e muovere giocatori e missili :

Istruzione MOVE
OPTION(PLM1 o PLM2)
VARPTR(PLM1 o PLM2)
SETCOLOR Ø o 1 o 2 o 3

LE ISTRUZIONI MICROSOFT BASIC II E LA GRAFICA SPECIALE

L'istruzione MOVE viene usata per spostare il giocatore o il missile in alto e in basso. Un foglio di carta può servire a dimostrare come funziona l'istruzione MOVE. Supponiamo di aver disegnato una V capovolta sul foglio di carta con una penna ad inchiostro cancellabile.Per spostare l'oggetto, bisogna cancellarlo tutto e ridisegnarlo in posizione diversa.
Come si può immaginare, il movimento verticale è leggermente più lento di quello orizzontale. E' più lento perchè basta una sola scrittura nel registro della posizione orizzontale per eseguire il movimento orizzontale, mentre per spostare un oggetto verticalmente sono necessarie molte cancellazioni e ridisegnature.
Nella istruzione MOVE si stabilisce l'indirizzo più basso assunto dall'oggetto e successivamente l'indirizzo più basso della nuova area in cui si vuole spostare l'oggetto; infine, si stabilisce quanti bytes si vogliono spostare . Da qui il formato : MOVE indirizzo-da, indirizzo-a, numero di bytes.

OPTION(PLM1) azzera e riserva nella memoria RAM un'area giocatore e missile con risoluzione a riga singola. OPTION(PLM2) esegue la stessa operazione con risoluzione a doppia riga.

VARPTR(PLM1 o PLM2) indica l'indirizzo iniziale di memoria dell'area giocatore e missile nella RAM. Questo è il punto da cui si deve valutare la distanza o lo spostamento per inserire l'immagine nell'area corretta. Per esempio, l'indirizzo iniziale (parte alta dello schermo) per il giocatore Ø in una soluzione a doppia riga è VARPTR(PLM2)+128. Nella risoluzione a doppia riga ogni giocatore è lungo 128 bytes. Perciò se si vuole tracciare una riga diritta davanti al giocatore Ø, si dovrà usare : POKE VARPTR(PLM2)+192, &FF.

L'istruzione SETCOLOR fornisce le assegnazioni di registro colore e luminosità. Nel Microsoft BASIC II Atari, i registri 0,1,2 e 3 vengono usati per i giocatori e missili 0,1,2 e 3. Per poter usare il giocatore e missile Ø, bisogna solo specificare SETCOLOR, 0, 5, 10. L'istruzione COLOR non viene usata.

Ricordare che si deve sempre memorizzare all'indirizzo decimale 559 il valore decimale 62 per la risoluzione a riga singola e il decimale 46

per la risoluzione a doppia riga. Inoltre, per abilitare la visualizzazione del giocatore e missile occorre memorizzare all'indirizzo decimale 53277 il decimale 3.

I grafici giocatore e missile possono essere usati in tutti i modi. I missili sono "strisce" larghe 2 bit. Ai missili 0. 1, 2, 3 è assegnato lo stesso valore del giocatore associato. Perciò, quando SETCOLOR stabilisce che il colore del giocatore 1 è rosso, anche il missile 1 sarà rosso.

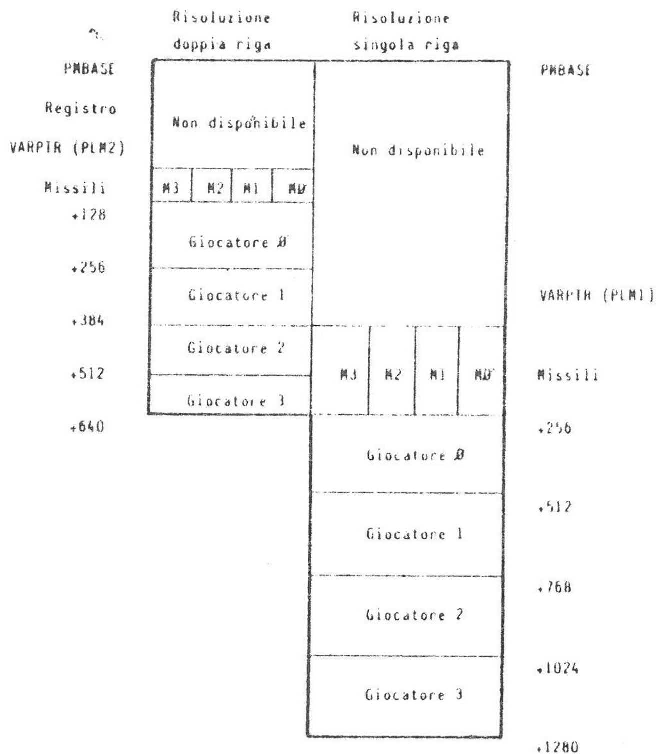


Fig. 7.1 - Configurazione della RAM dei Grafici giocatore e missile

☐ SIMULAZIONE DI UN GIOCATORE

Tagliare una striscia di carta larga circa 5 cm e lunga 25. Poi disegnare un "byte" largo 8 bit per la lunghezza della striscia.

Rappresentazione GRAFICA	Rappresentaz. binaria	Rappresentaz. esadecimale	Rappresentaz. decimale
	00011000	18	24
	00011000	18	24
	00100100	24	36
	00100100	24	36
	01000010	42	66
	01000010	42	66
	10000001	81	129
	10000001	81	129

Figura 7-2 Mappa di un giocatore

Sulla striscia di carta è mostrata una V capovolta con i valori in binario e in esadecimale. Questa striscia di carta corrisponde ad un giocatore. Se si colloca la striscia-giocatore verticalmente in mezzo allo schermo, questa è stata "posizionata" con il registro di posizione orizzontale.
Quando si muove la striscia a destra e a sinistra è come se venissero introdotti (POKE) nuovi indirizzi nel registro di posizione orizzontale per ottenere tale movimento.

☐ CONTROLLO DEL COLORE

Il calcolatore ATARI ha nove registri che permettono all'utente di controllare il colore del missile-giocatore, del campo di gioco e dello sfondo (vedere Tabella 7-1).

SETCOLOR registro, colore, luminosità	Funzione
SETCOLOR Ø, colore, luminosità	Colore-luminosità del missile-giocatore Ø
SETCOLOR 1, colore, luminosità	Colore-luminosità del missile-giocatore 1
SETCOLOR 2, colore, luminosità	Colore-luminosità del missile-giocatore 2
SETCOLOR 3, colore, luminosità	Colore-luminosità del missile-giocatore 3
SETCOLOR 4, colore, luminosità	Colore-luminosità del campo gioco Ø
SETCOLOR 5, colore, luminosità	Colore-luminosità del campo gioco 1
SETCOLOR 6, colore, luminosità	Colore-luminosità del campo gioco 2
SETCOLOR 7, colore, luminosità	Colore-luminosità del campo gioco 3
SETCOLOR 8, colore, luminosità	Colore-luminosità dello sfondo

Tabella 7-1 Assegnazioni del registro del colore

I giocatori sono completamente indipendenti tra di loro e dal campo di gioco. I missili associano i registri colore con i loro giocatori e perciò hanno lo stesso colore del giocatore loro associato. Se si uniscono i missili per formare un quinto giocatore, essi assumono il colore del registro 3 (COLPF3).

Per ottenere un colore particolare occorre specificare il registro, la tonalità del colore e la luminosità, utilizzando il comando SETCOLOR. Vedere le righe 20 e 110 del programma dimostrativo per i grafici giocatore e missile per avere degli esempi. Vedere anche il Capitolo 6. Ogni registro di colore-luminosità è indipendente ed è perciò possibile usare in un programma fino a nove colori diversi a seconda del modo grafico scelto. Non tutti i registri però possono essere usati in tutti i modi grafici.

☐ CONTROLLO DELLA DIMENSIONE

Vengono forniti cinque registri di controllo della dimensione. Quattro per i giocatori e uno per tutti e quattro i missili. Vedere Tabella 7-2.

Registro dimensione	Indirizzo		Funzione
	Esadecim.	Decimale	
SIZEP0	D008	53256	Controlla la dimensione del giocatore 0
SIZEP1	D009	53257	Controlla la dimensione del giocatore 1
SIZEP2	D00A	53258	Controlla la dimensione del giocatore 2
SIZEP3	D00B	53259	Controlla la dimensione del giocatore 3
SIZEM	D00C	53260	Controlla la dimensione dei missili

Tabella 7-2 Registri che controllano la dimensione del giocatore e del missile

I registri che controllano la dimensione permettono di raddoppiare o quadruplicare la dimensione di un giocatore o di un missile senza alterare la sua risoluzione in bit. Per raddoppiare le dimensioni memorizzare tramite POKE un 1 nel registro della dimensione; per quadruplicarla, memorizzare tramite POKE un 3 e per riportare un giocatore o un missile alla dimensione normale, memorizzare uno Ø oppure 2.

Un esempio viene fornito nella pagina 80 del Programma Dimostrativo per grafica del giocatore e missile.

[.] POSIZIONE E MOVIMENTO

- Verticale

La posizione verticale viene stabilita quando si specifica la poszione del giocatore e missile nella memoria RAM della grafica. Tanto più in basso viene posto nella RAM il giocatore e missile, tanto più in alto appare l'immagine nello schermo televisivo.

Una tecnica di posizionamento é illustrata nelle righe 120 e 200 del Programma dimostrativo per grafica, inserito alla fine di questo capitolo.

Per programmare il movimento verticale, usare il comando MOVE. Vedere le righe 350 e 390 del Programma dimostrativo. Poichè il comando MOVE non azzera la vecchia posizione dopo il trasferimento, in coda ad ogni giocatore viene aggiunto uno zero aggiuntivo per "pulire" la precedente posizione non appena il giocatore è stato spostato. Occorre inoltre fornire la posizione attuale del giocatore in memoria RAM, la direzione del movimento nella RAM (avanti = +, indietro = -) ed il numero totale dei bytes che rappresentano il giocatore che devono essere trasferiti.

Dopo il comando MOVE, occorre incrementare o decrementare il contatore della posizione verticale. Vedere le righe 360 e 400 del Programma dimostrativo.

- Orizzontale

Ogni giocatore e ogni missile ha un suo registro relativo alla posizione orizzontale (Vedere Tabella 7-3); ciò permette spostamenti totalmente indipendenti gli uni dagli altri; i missili pure si possono spostare in modo indipendente rispetto ai loro giocatori.

Registro posizione	Indirizzo Esadec.	Indirizzo Dec.	Funzione
HPOSP0	D000	53248	Posizione orizzontale del giocatore 0
HPOSP1	D001	53249	Posizione orizzontale del giocatore 1
HPOSP2	D002	53250	Posizione orizzontale del giocatore 2
HPOSP3	D003	53251	Posizione orizzontale del giocatore 3
HPOSM0	D004	53252	Posizione orizzontale del missile 0
HPOSM1	D005	53253	Posizione orizzontale del missile 1
HPOSM2	D006	53254	Posizione orizzontale del missile 2
HPOSM3	D007	53255	Posizione orizzontale del missile 3

Tabella 7-3 Registri posizione orizzontale giocatore e missile

Per stabilire la posizione di un giocatore o di un missile, trasferire (POKE) nel suo registro relativo alla posizione orizzontale il numero della posizione stessa. Per programmare il movimento orizzontale, basta modificare il numero memorizzato nel registro. Vedere le righe 100 e 180 del Programma Dimostrativo per avere degli esempi.

Un registro di posizione orizzontale può contenere fino a 256 posizioni, ma alcune di esse si trovano fuori del margine sinistro o destro dello schermo televisivo. Una stima di sicurezza dell'arco visivo del giocatore è valutata dalla posizione orizzontale 60 alla posizione 200. L'arco visivo reale dipende anche molto dal tipo di televisore.

- Diagonale

I movimenti orizzontale e verticale possono essere combinati assieme per smuovere il giocatore in senso diagonale. Prima bisogna stabilire la posizione orizzontale e poi quella verticale. Esempi vengono forniti nelle righe da 270 a 390 del Programma dimostrativo.

CONTROLLO PRIORITA'

Il registro di controllo della priorità (PRIOR, &D01B; OS ombra del GPRIOR, &26F) permette di selezionare la priorità del registro del colore del giocatore o del campo di gioco e di unire i missili per formare un quinto giocatore.

- Selezione della priorità

Si ha la possibilità di specificare quale immagine sia prioritaria nel caso in cui le immagini del giocatore e del campo di gioco si sovrappongano.

Questa funzione permette di far scomparire i giocatori dietro il campo di gioco e viceversa. Per stabilire la priorità, trasferire (POKE) uno dei seguenti numeri nel registro di controllo della priorità.

1 = Tutti i giocatori hanno priorità su tutti i campi da gioco.
2 = I giocatori Ø e 1 hanno priorità su tutti i campi da gioco e tutti i campi da gioco hanno priorità sui giocatori 2 e 3.
4 = Tutti i campi da gioco hanno priorità su tutti i giocatori.
8 = I campi da gioco Ø e 1 hanno priorità su tutti i giocatori e tutti i giocatori hanno priorità sui campi da gioco 2 e 3.

• Abilitazione del quinto giocatore

Se si pone ad 1 il bit D4 del registro di controllo della priorità, tutti i missili assumono il colore dato dal registro 3 del campo di gioco (&2C7, decimale 711). I missili possono essere uniti per formare un quinto giocatore. Se ciò viene eseguito, il quinto giocatore deve essere ancora mosso orizzontalmente, modificando tutti i registri dei missili (da &D004 a &D007) simultaneamente.

☐ CONTROLLO COLLISIONE

Il controllo collisione permette di evidenziare quando un giocatore o un missile sia entrato in collisione con un altro oggetto. Esistono 16 registri di controllo-collisione. Vedere Tabella 7-4.

Registro collisione	Indirizzo Esadec.	Decim.	Funzione
M0PF	D000	53248	Missile Ø a campo di gioco
MAPF	D001	53249	Missile 1 a campo di gioco
M2PF	D002	53250	Missile 2 a campo di gioco
M3PF	D003	53251	Missile 3 a campo di gioco
P0PF	D004	53252	Giocatore Ø a campo di gioco
P1PF	D005	53253	Giocatore 1 a campo di gioco
P2PF	D006	53254	Giocatore 2 a campo di gioco
P3PF	D007	53255	Giocatore 3 a campo di gioco
M0PL	D008	53256	Missile Ø a giocatore
M1PL	D009	53257	Missile 1 a giocatore
M2PL	D00A	53258	Missile 2 a giocatore
M3PL	D00B	53259	Missile 3 a giocatore
P0PL	D00C	53260	Giocatore Ø a giocatore
P1PL	D00D	53261	Giocatore 1 a giocatore
P2PL	D00E	53262	Giocatore 2 a giocatore
P3PL	D00F	53263	Giocatore 3 a giocatore

Tabella 7-4 Registri controllo-collisione giocatore e missile

In ogni caso, vengono usati solo i 4 bits più a destra di ogni registro. Essi occupano le posizioni 0,1,2 e 3 a partire da destra ed indicano attraverso la loro posizione, con quale campo di gioco o giocatore il giocatore o il missile in questione sia entrato in collisione.

Il valore 1 in qualsiasi bit indica la collisione dall'ultimo HITCLR. Tutti i registri di collisione vengono azzerati simultaneamente scrivendo uno zero sul registro HITCLR (&D01E, decimale 53278).

PROGRAMMA DIMOSTRATIVO DI GRAFICA GIOCATORE E MISSILE CON DOPPIA RISOLUZIONE

Il seguente programma Atari Microsoft Basic II genera un giocatore (navicella spaziale) che lancia missili e che può essere mosso in ogni direzione per mezzo del comando a cloche.
Ricordate di collegare il governo a cloche al vostro computer Atari.

```
5 !GRAFICA GIOCATORE E MISSILE CON DOPPIA RISOLUZIONE
10 GRAPHICS 8
20 SETCOLOR 6,0,0
30 X=180
40 Y=70
50 STICK0=&278
60 OPTION PLM2
70 POKE 559,46
80 POKE &D00C,1
90 POKE &D01D,3
100 POKE &D000,X
110 SETCOLOR 0,3,10
120 FOR J=VARPTR(PLM2)+128+Y TO VARPTR(PLM2)+135+Y:READ A:POKE J,A
125 NEXT J
130 DATA 0,129,153,189,255,189,153,0
140 IF PEEK(&D010)=1 THEN 220
150 SOUND 0,220,12,15,INT(X/30)
160 ZAP= X
170 POKE VARPTR(PLM2)+4+Y,3
180 POKE &D004,ZAP
190 ZAP=ZAP-12
200 IF ZAP<12THEN POKE VARPTR(PLM2)+4+Y,0:GOTO 220 ELSE 180
210 !MOVIMENTI DEL JOYSTICK
220 A=PEEK(STICK0):IF A=15 THEN GOTO 140
230 IF A=11 THEN X=X-1
240 IF A=7 THEN X=X+1
250 POKE &D000,X
260 IF A=14 THEN GOTO 350!MOVIMENTO IN ALTO
270 IF A=13 THEN GOTO 390!MOVIMENTO IN BASSO
280 !MOVIMENTO DIAGONALE
290 IF A=10 THEN X=X-1:POKE &D000,X:GOTO 350
300 IF A=6 THEN X=X+1:POKE &D000,X:GOTO 350
310 IF A=9 THEN X=X-1:POKE &D000,X:GOTO 390
320 IF A=5 THEN X=X+1:POKE &D000,X:GOTO 390
330 GOTO 140
340 ! SPOSTAMENTO VERTICALE
350 MOVE VARPTR(PLM2)+128+Y,VARPTR(PLM2)+128+(Y-1),8
360 Y=Y-1
370 GOTO 140
380 !MOVIMENTO IN BASSO
390 MOVE VARPTR(PLM2)+128+(Y-1),VARPTR(PLM2)+128+Y,8
400 Y=Y+1
410 GOTO 140
420 STOP
430 END
```

ANNOTAZIONI

Numero di riga	Commenti
10	Abilita un modo grafico ad alta risoluzione senza nessuna area di testo. Si possono programmare grafici giocatore e missile in qualsiasi modo grafico. Vedere il paragrafo "GRAFICI" e la Tabella 6-4 al Capitolo 6.
20	Colora di nero lo sfondo nel seguente modo : 6 - Registro colore-luminosità dello sfondo (COLBK, &D01A) 0 = Nero (Vedere Tabella 6-3). 0 - Luminosità zero. Il valore di luminosità è un numero pari compreso tra 0 e 14. Più elevato è il numero, maggiore è la luminosità e più brillante il colore.
30,40	Inizializza le coordinate X (orizzontale) e Y (verticale) della posizione del giocatore.
50	Assegna la label STICKO al registro del governo a cloche 278.
60	Specifica la memoria RAM con risoluzione a doppia riga per i grafici giocatore e missile. vedere Fig. 7-1. PLMI specifica la risoluzione a riga singola.
70	Abilita il registro di controllo dell'accesso diretto in memoria (DMACTL, 559) per la risoluzione a riga doppia (46). Il numero 62 indica invece la risoluzione a riga singola. Nota : Quando si abilita DMACTL i registri dei grafici giocatore e missile (GRAFP0-GRAFP3 e GRAFM) vengono automaticamente caricate con i dati provenienti dalla RAM giocatore e missile.
80	Raddoppia la larghezza del missile andando a mettere 1 (POKE) nel registro di controllo della dimensione (SIZEM, &D00C). Un 3 quadruplicherebbe la larghezza.
90	Abilita il registro di controllo della grafica (GRACTL, &D01D) per visualizzare la grafica giocatore e missile (3 abilita, Ø disabilita).
100	Trasferisce (POKE) la posizione orizzontale del giocatore (X - 130 da riga 30) nel registro di posizione orizzontale del giocatore 0 (HPOSP0, &D000).
110	Colora il giocatore e il missile in rosso-arancio brillante come segue: 0 - registro colore-luminosità del missile-giocatore 0 (COLPM0, &D012).

3 = Rosso-arancio (vedere tabella 6-3)
10 = Luminosità e brillantezza (vedi automazione della riga 20).

120-125 Posiziona il puntatore della variabile VARPTR(PLM2) sull'indirizzo iniziale del giocatore e missile nella RAM della grafica giocatore e missile . Vedere Figura 7-2. Trasferisce tramite POKE i dati dalla riga 130 nell'area dei giocatori e VARPTR(PLM2) +128+Y a VARPTR(PLM2) +135+Y. Il calcolatore utilizza i dati della riga 130 per realizzare visivamente la nave spaziale sullo schermo. Vedere Figura 7-2.

140 Comanda al calcolatore di leggere il registro del grilletto associato al comando a cloche Ø (TRIGO, &D010). Se il grilletto non viene azionato (&D010=1), l'esecuzione passa alla riga 200 e legge la posizione del comando a cloche. Se invece il grilletto è stato azionato (&D010=0), il calcolatore esegue le righe da 150 a 200.

150 Genera un suono ogni volta che si preme il grilletto della cloche. Tale suono è programmato come segue :

1) Selezionare la voce. Si possono usare fino a 4 voci (da 0 a 3), ma ogni voce richiede uno statement SOUND separato.
2) Scegliere la tonalità dalla Tabella 7-2. Più alto è il numero, più bassa è la tonalità.
3) Stabilire il livello di distorsione o di rumore, usando un numero pari compreso tra 0 e 14. Un 10 da un tono puro; 12 da un effetto di cicalino.
4) Stabilire il volume, un numero dispari compreso tra 1 e 15. Più alto è il numero, più forte il suono.
5) Stabilire la durata del suono in secondi (20 = 20/60 o 1/3 di secondo).

160 Pone la posizione orizzontale del missile (ZAP) uguale alla posizione orizzontale del giocatore (X).

170 Abilita i pixels dello schermo corrispondenti all'area di RAM del missile 0 (VARPTR(PLM2)+4+Y) per visualizzare il missile (3 = ON; 0 = OFF).

180 Trasferisce con POKE la posizione orizzontale del missile (ZAP = X dalla riga 160) nel registro della posizione orizzontale del missile 0 (HPOSMO, &D004).

190 Decrementa il contatore della posizione orizzontale del missile 0 di 12 per creare una "riga di fuoco" orizzontale davanti al giocatore.

200 Se la posizione orizzontale è minore di 12 (fuori dal lato sinistro dello schermo) il calcolatore trasferisce con POKE degli "0" nell'area RAM del missile per ripulirla ed il programma prosegue alla riga 220.

Se la posizione orizzontale del missile è 12 o maggiore di 12, il calcolatore trasferisce la nuova posizione orizzontale in HPOSMO (registro &D004 nella riga 180) e decrementa di 12 il contatore della posizione orizzontale (riga 190).

220 Comanda al calcolatore di leggere il registro STICKO e di cercare la posizione della cloche.(Vedere Figura 6-1). Se la posizione è 15 (neutra), il programma salta alla riga 140 e va a leggere il registro del grilletto della cloche (&D010).

230/250 Se la cloche viene spostata verso sinistra (11), il calcolatore decrementa il contatore della posizione orizzontale e trasferisce la nuova posizione orizzontale della nave spaziale nel registro HPOSPO (&D000) tramite POKE.

240/250 Se la cloche viene spostata verso destra (7), il calcolatore incrementa il contatore della posizione orizzontale e trasferisce la nuova posizione orizzontale della nave spaziale in HPOSPO.

260 Se la cloche viene spostata verso l'alto (14), il calcolatore muove la nave spaziale indietro di un byte nella RAM dedicata al giocatore e missile (riga 350). Ognuno degli 8 bytes che formano la nave spaziale deve essere spostato indietro. Una volta completato il movimento, il calcolatore decrementa il contatore della posizione verticale (riga 360).

270 Se la cloche viene spostata verso il basso (13), il calcolatore fa avanzare la nave spaziale di un byte nella RAM dedicata al giocatore e missile (riga 390) ed incrementa il contatore della posizione verticale (riga 400).

290-320 Se la cloche viene spostata in diagonale (10,6,9 o 5), il calcolatore esegue un movimento orizzontale (dopo aver riposizionato il registro della posizione orizzontale), un movimento verticale (riga 350 o 390) e riposiziona il contatore della posizione verticale (righe 360 o 400).

Appendice A
Programmi Esempio

PROGRAMMA PER LEGGERE LA DIRECTORY DA BASIC

Caratteristiche usate:

- Utilizzo della routine CIO (CIOUSR) (Vedi Appendice L)
- Interi
- Funzione VARPTR
- ON ERROR
- Commenti sulle linee di programma.

```
1 LPRINT "CANE"
2 STOP
10 !                              ROUTINE PER LEGGERE
20 !                              IL DISK DIRECTORY
30 !
40 ON ERROR 350
50 OPTION RESERVE(200)            !RISERVA SPAZIO IL CIOUSR
60 OPEN #1,"D:CIOUSR" INPUT       !APRE IL FILE
80 ADDR=VARPTR(RESERVE)           !PRENDI L'INDIRIZZO INIZILE DELL'AREA
90 FOR I=0 TO 159                 !POKE LA CIOUSR
100 GET #1,D:POKE ADDR+I,D
110 NEXT I
120 CLOSE #1
130 PUTIOCB=ADDR                  !QUESTI SONO I PUNTI DI INIZ_O
140 CALLCIO=ADDR+61               !PER OGNUNA DELLE
150 GETIOCB=ADDR+81               !ROUTINE
160 DIM IOCB%(10)                 !I DATI PER LA ROUTINE USANO 10 BYTES
170 IOCB%(0)=1                    !USE IOCB #1
180 IOCB%(1)=3                    !FAI UNA CIO "OPEN" CALL
190 IOCB%(5)=6                    !PERIMMETTERE LA DIRECTORY
200 FSPEC$="D:*.*"                !SPECIFICA IL FILE DIR
210 !              METTI L'INDIRIZZO DI FSPEC NEL O BUFFER DEL IOCB
220 Z=VARPTR(FSPEC$)              !INDIRIZZO DELLA STRINGA FILESPEC
230 Y=VARPTR(IOCB%(3))            !INDIRIZZO DELLA POSIZIONE DELLA MATRICE
240 POKE Y,PEEK(Z+2)              !BYTE DI INDIRIZZO PIU' SIGNIFICATIVO
250 POKE Y+1,PEEK(Z+1)            !E MENO SIGNIFICATIVO
260 !                             METTI I DATI NELLO IOCB
270 Z=USR(PUTIOCB,VARPTR(IOCB%(0)))
280 !                         E CHIAMA IL CIO
290 Z=USR(CALLCIO,VARPTR(IOCB%(0)))
300 !                             IOCB E' OK E IL DISCO
310 !                             E' APERTO E ...VIA !!!!!
320 INPUT #1,S$
330 PRINT S$
340 GOTO 320
350 CLOSE #1
360 END
```

SUBROUTINE PER GENERARE UN'ESPLOSIONE

Caratteristiche usate: Suono.

```
10 ! UN PROGRAMMA DI SOLE DUE LINEE
20 ! ED UNA BELLA SUBROUTINE
30 ! PER UNA ESPLOSIONE
50 GOSUB 8000
60 STOP
8000 !
8010 ! ESPLOSIONE !!!!!
8020 !
8030 SOUND 2,75,8,14
8040 ICR=0.79
8050 V1=15:V2=15:V3=15
8060 SOUND 0,NTE,8,V1
8070 SOUND 1,NTE+20,8,V2
8080 SOUND 2,NTE+50,8,V3
8090 V1=V1*ICR
8100 V2=V2*(ICR+.05)
8110 V3=V3*(ICR+.08)
8120 IF V3=>1 THEN 8060
8130 SOUND 0,0,0,0,0
8140 SOUND 1,0,0,0,0
8150 SOUND 2,0,0,0,0
8160 RETURN
```

ESEMPIO DI FANFARA

Caratteristiche usate: Suoni prolungati.

```
10 ! UN PROGRAMMA DI SOLE DUE LINEE
20 ! ED UNA BELLA SUBROUTINE
30 ! PER UNA FANFARA
40 GOSUB 8000
50 STOP
8000 !
8010 ! FANFARA    !!!!:
8020 !
8030 DUR=20:VO=181:V1=144:V2=121:GOSUB 8200
8040 DUR=7 :GOSUB 8200
8050 GOSUB 8200
8060 DUR=9:VO=162:V1=128:V2=108:GOSUB 8200
8070 DUR=15:VO=181:V1=144:V2=121:GOSUB 8200
8080 VO=162:V1=128:V2=108:GOSUB 8200
8090 VO=153:V1=128:V2=96:V3=193
8100 FOR I=2 TO 14
8110 SOUND 3,VO,10,I
8120 SOUND 1,V1,10,I
8130 SOUND 2,V2,10,I
8140 SOUND 0,V3,10,I
8150 FOR J=1 TO 100 :NEXT J
8160 NEXT I
8170 FOR J=1 TO 200:NEXT J
8180 SOUND 0,0,0,0,0
8185 SOUND 1,0,0,0,0
8190 SOUND 2,0,0,0,0
8195 SOUND 3,0,0,0,0
8197 RETURN
8200 !GENERATORE SUONI
8210 SOUND 0,VO,10,8,DUR
8220 SOUND 1,V1,10,8,DUR
8230 SOUND 2,V2,10,8,DUR
8240 REM
8270 SOUND 0,0,0,0,0
8280 SOUND 1,0,0,0,0
8290 SOUND 2,0,0,0,0
8295 FOR J=1 TO 250:NEXT J
8300 RETURN
```

ESEMPIO DI PIANOFORTE

Caratteristiche usate:

- Istruzione OPEN
- Matrice di stringa
- Istruzione INKEY$
- Suono
- Commenti sulle linee di programma.

```
10 ! ESEMPIO DI PROGRAMMA
20 ! PER TRASFORMARE IL TUO
30 ! ATARI IN UN PIANOFORTE
40 !
50 !
60 ! DEFINISCI UNA SCALA DI
70 ! 2 OTTAVE PER I TASTI
80 ! E PER LE NOTE
90 DIM NOTES$(15)
100 DIM PITCH(15)
110 ! ORA ASSEGNA I VALORI
120 ! RISPETTIVI
130 OPEN #1,"D1:NOTES.DAT"INPUT
140 FOR I =1 TO 15
150 INPUT #1,S$,P
160 NOTES$(I)=S$:PITCH(I)=P
170 NEXT I
180 CLOSE #1
190 PRINT"SUONA ....!!!!!!"
200 !
210 !PROVA CON I TASTI
220 !SUONA......
230 !
240 N$=INKEY$
250 IF N$=""THEN GOTO 240 ELSE GOTO 320
260 !
270 ! SE UN TASTO E PREMUTO
280 ! CONTROLLA SE FA PARTE
290 !  DEL TUO PIANOFORTE
300 !
310 !
320 FOR I= 1 TO 15
330 IF N$-NOTES$(I) GOTO 380
340 NEXT I
350 GOTO 240 !
360 ! E' UNA CHIAVE GIUSTA
370 ! FAI IL TUO DOVERE !!!
380 VOLUME =8
390 SOUND 1,PITCH(I),10,VOLUME,15
400 GOTO 240
410 END
```

PROGRAMMA PER GENERARE IL FILE DI NOTE:

- Prima assegna il tasto
- Poi la relativa nota.

```
10 ! PROGRAMMA PER CREARE IL FILE
20 ! NOTES.DAT DI NOTE
30 DIM NOTES$(15),PITCH(15)
40 FOR I= 1 TO 15
50 INPUT"ASSEGNA IL TASTO E LA NOTA :";NOTES$(I),PITCH(I)
60 NEXT I
70 OPEN #1,"D1:NOTES.DAT"OUTPUT
80 FOR I=1 TO 15
90 PRINT #1,NOTES$(I);",";PITCH(I)
100 NEXT I
110 CLOSE #1
120 END
```

Introdurre i seguenti valori per ottenere una scala di due ottave:

```
Z, 243
X, 217
C, 193
V, 182
B, 162
N, 144
M, 128
A, 121
S, 108
D, 96
F, 91
G, 81
H, 72
J, 64
K, 60
```

PROGRAMMA PER LA CONVERSIONE DEI NUMERI DECIMALI IN ESADECIMALI

Caratteristiche usate:

- Matrice di stringa
- Interi
- Commenti sulle linee di programma.

```
20 !
30 !DECHEX
40 !
50 !
60 !
70 !PROGRAMMA PER CONVERTIRE
80 !UN NUMERO DECIMALE NEL
90 !SUO EQUIVALENTE ESADECIMALE
100 !
110 !
120 !
130 DIM HEX$(15):DIM HEXBASE(4)
140 FOR I=0 TO 15
150 READ HEX$(I)
160 NEXT I
170 FOR I= 0 TO 4
180 READ HEXBASE(I)
190 NEXT I
200 DATA 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F
210 DATA 0,4096,256,16,1
220 !
230 ! IMMISSIONE NUMERO DECIMALE
240 !
250 INPUT "SCRIVI IL NUMERO DECIMALE :";DEC
260 IF DEC=0 THEN 500 !STOP
270 !
280 !CONVERSIONE IN ESADECIMALE
290 !
300 FOR J=1 TO 4
305 IF J=4 THEN ANS%=DEC:GOTO 350
310 ANS%=(DEC/HEXBASE(J))-.5
320 IF ANS%<1 THEN ANS%=0
330 DEC=DEC-(ANS%*HEXBASE(J))
340 !
350 !PRIMO VALORE ESADECIMALE
360 FOR I%=0 TO 15
370 IF ANS%=I% THEN GOTO 420
380 NEXT I%
390 !VERIFICA
400 PRINT"VALORE DECIMALE NON VALIDO"
410 PRINT"PROVA ANCORA":GOTO 250
420 HEXNO$=HEXNO$+HEX$(I%)
430 NEXT J
440 !
450 !STAMPA DEL VALORE ESADECIMALE
460 !IMMISSIONE DI UN NUOVO NUMERO
470 PRINT"VALORE ESADECIMALE = ";HEXNO$
480 HEXNO$=""
490 GOTO 250
500 END
```

PROGRAMMA PER SCROLLING VERTICALE

Caratteristiche usate:

- Scrolling
- VARPTR
- OPTION RESERVE e CHR
- DISPLAY LIST.

```
10 DEFINT A-Z
20 OPTION RESERVE(3000)    !RAM RISERVATA AL VIDEO
30 OPTION CHR1             !DISPLAY LIST
40 ADDR=VARPTR(CHR1)
50 CADDR=VARPTR(RESERVE)
60 VSCROL=&D405            !REGISTRO SCROLL VERTICALE
70 LCADDR=0
80 HCADDR=((CADDR AND &FF00)/256) AND  &FF
90 FOR I= 0 TO 99          !AZZERA I PRIMI CENTO BYTE
100 POKE ADDR+I,0:NEXT I   ! DELLA AREA DISPLAY LIST
110 LADDR=ADDR AND &FF
120 HADDR=((ADDR AND &FF00)/256)AND &FF
130 LMSLO=ADDR+4           ! INDIRIZZO DEI BYTES DI SCAN
140 LMSHI=ADDR+5           ! DELLA MEMORIA
150 FOR I =0 TO 18         ! POKE DELLA NUOVA DISPLAY LIST
160 READ D                 ! DAI DATA
170 POKE ADDR+I,D
180 NEXT I
190 DATA &70,&70,&70,&67,&00,&00,&27,&27
200 DATA &27,&27,&27,&27,&27,&27,&27,&27
210 DATA &27,&07,&41
220 POKE ADDR+19,LADDR     ! GLI ULTIMI DUE BYTES PUNTANO
230 POKE ADDR+20,HADDR     ! ALLA CIMA DELLA DISPLAY LIST
240 POKE LMSLO,LCADDR:POKE LMSHI,HCADDR  ! INIZIO DELLA RAM DI SCHERMO
250 K=-1
260 FOR I =1 TO 300
270 K=K+1:POKE CADDR+K,33
280 NEXT I
290 FOR I = 34 TO 58
300 FOR J=1 TO 20
310 K=K+1:POKE CADDR+K,I
320 NEXT J,I
330 POKE &22F,0    ! SPEGNI ANTIC
340 POKE &230,LADDR ! DIGLI DOVE E' LA MIA
350 POKE &231,HADDR  ! DISPLAY LIST E ...
360 POKE &22F,&22 ! RIACCENDILO
370 !  ECCO IL VERO PROGRAMMA
380 FOR I= 1 TO 15   ! SCROLL FINE !!!
390 POKE VSCROL,I
400 FOR W=1 TO 30:NEXT W
410 NEXT I
420 CADDR = CADDR+20 !
430 LCADDR=CADDR AND &FF
440 HCADDR=((CADDR AND &FF00)/256) AND &FF
450 WAIT &D40B,&FF,96
460 POKE VSCROL,0
470 POKE LMSLO,LCADDR
480 POKE LMSHI,HCADDR
490 GOTO 380
```

Appendice B
Programmi Grafici

PROGRAMMI PER MODI GRAFICI

Invasione di microbi.

```
10 ! INVASIONE DEI MICROBI
15 ! CREATURE INVADONO LO SCHERMO
16 !
17 !
20 ! E QUANDO ESCONO DALLO SCHERMO ...
30 RANDOMIZE
40 MODE=RND(8)
50 GRAPHICS MODE+16
60 PIX= RND(15)
70 SETCOLOR  0,PIX,6
80 COLOR 1
90 BAK=RND(255)
100 POKE 712,BAK
110 X=RND(150):Y=RND(100)
120 IF X>140 THEN 40
130 Z=2
140 NUM=NUM+1
150 FOR DOTS=1 TO Z
160 IF NUM =5 THEN NUM =1
170 ON ERROR GOTO 230
180 PLOT X,Y
190 ON NUM GOSUB 250 ,270, 290, 310
200 NEXT
210 Z=Z + 1
220 GOTO 140
230 GRAPHICS MODE + 32+16
240 RESUME 60
250 X=X+1:Y=Y+1
260 RETURN
270 X=X+1:Y=Y-1
280 RETURN
290 X=X-1:Y=Y-1
300 RETURN
310 X=X-1:Y=Y+1
320 RETURN
```

PROGRAMMA TOP SECRET

Il seguente breve programma utilizza RANDOMIZE e AND per stampare parole di tre lettere ed abbreviazioni di tre lettere delle Agenzie Governative.

```
10 RANDOMIZE            ! PONI IL SEME ALLA RND
20 GRAPHICS 2+16
30 X=RND(26)+96         ! FAI LA PRIMA LETTERA
40 Y=RND(5)             ! SCEGLI UNA VOCALE
50 IF 1=1 THEN Y=97     ! UNA A
60 IF Y=2 THEN Y=101    ! UNA E
70 IF Y=3 THEN Y=105    ! UNA I
80 IF Y=4 THEN Y=111    ! UNA O
90 IF Y=5 THEN Y=117    ! UNA U
100 Z=RND(26)+96        ! ULTIMA LETTERA
110 PRINT #6,AT(9,3)CHR$(X);CHR$(Y);CHR$(Z)
120 FOR RITARDO=1 TO 2000:NEXT
130 GOTO 30
```

Appendice C
Set di Caratteri Alternativi

GENERALITA'

Gli Home Computer ATARI supportano alcuni insiemi di caratteri standard che sono memorizzati come parte della ROM del Sistema Operativo (S.O). Tali insiemi comprendono tutte le lettere dell'alfabeto maiuscole e minuscole, i numeri, i caratteri speciali ed uno speciale gruppo di caratteri grafici.
Tuttavia, può risultare utile essere in grado di generare caratteri e segni personalizzati per soddisfare esigenze connesse ad animazioni, word processing di lingue straniere e grafici di sfondo per i giochi (per esempio una mappa o un campo di gioco speciale).
I calcolatori Atari ed il Microsoft BASIC II Atari risolvono facilmente queste esigenze, dato che il data base del S.O. contiene un puntatore (CHBAS) alla locazione esadecimale 2F4 (decimale 756) che punta al set di caratteri da usare.
Normalmente tale puntatore indica l'insieme di caratteri standard nella ROM del S.O. però, utilizzando il BASIC, si può trasferire tramite POKE l'insieme di caratteri personalizzato in un'area libera della memoria RAM (riservata con lo statement OPTION CHR1 o OPTION CHR2) e poi ripristinare il puntatore O.S., CHBAS, in modo che indichi il nuovo insieme di caratteri. Immediatamente il calcolatore inizia ad usare i nuovi caratteri come standard.

Qui di seguito vengono elencati dei suggerimenti operativi relativi all'utilizzo di un insieme di caratteri o simboli personalizzati:

- Il modo grafico 0 richiede che siano definiti 128 caratteri (OPTION CHR1). I modi grafici 1 e 2 ammettono solo 64 caratteri (OPTION CHR2).
- Tutti i 64 o 128 caratteri devono essere definiti anche se si vuole modificarne ed usarne uno solo: ciò può essere fatto molto facilmente trasferendo i caratteri ROM nell'area di RAM e poi modificando il carattere voluto in modo che assuma la nuova configurazione.
- L'insieme di 64 caratteri richiede 512 bytes di memoria (8 bytes per carattere) e deve iniziare in una locazione di memoria multiplo di ½K. L'insieme di 128 caratteri richiede 1024 bytes di memoria e deve iniziare su un multiplo di 1K. Non bisogna preoccuparsi per queste restrizioni quando si usano le opzioni CHR1 e CHR2, dato che l'area

viene assegnata in modo tale che abbia inizio nel punto giusto di memoria.

- Il valore trasferito in CHBAS, dopo la definizione dell'insieme di caratteri, è il numero di pagina in memoria in cui iniziano i caratteri. Questo valore può essere calcolato con il seguente statement
  [illegible]
  [illegible]
  [illegible]

Senza dubbio la generazione e definizione di caratteri alternativi è l'operazione che richiede la maggior quantità di tempo.
Ogni carattere è formato da 8 bytes di memoria impilati uno sull'altro (vedere Figura C-1). Visualizzare ogni carattere come un quadrato 8 x 8 su carta millimetrata. Annerire il quadrato richiesto sulla carta millimetrata per creare un carattere (vedi Figura C-2).Poi ogni riga del quadrato 8 x 8 viene convertita da questa rappresentazione binaria (dove ogni quadrato annerito è 1 e ogni quadrato bianco è uno 0) ad un numero esadecimale o decimale (vedere Figura C-2). Tali numeri vengono poi scritti negli appropriati bytes dell'area RAM, dall'alto in basso di tali figure, in modo da definire il carattere in RAM.

I primi 8 bytes dell'area riservata (OPTION CHR1 o CHR2) definiscono il carattere zero, i successivi 8 bytes definiscono il primo carattere e così via. Dopo aver trasferito il set di caratteri standard dalla sua posizione nella ROM nell'area riservata CHR1 o CHR2, si può ridefinire qualsiasi carattere cercando la sua posizione iniziale nell'area e poi trasferire tramite POKE i bytes che descrivono il carattere nel byte di inizio e nei successivi 7 bytes.

Una volta che sono stati ridefiniti tutti i caratteri necessari, occorre trasferire il nuovo numero di pagina in CHBAS ed il nuovo set di caratteri viene reso immediatamente disponibile.

Volendo visualizzare i nuovi caratteri usare lo statement PRINT. Per esempio, se è stata ridefinita la "A" per trasformarla in un blocco solido e si è usato lo statement PRINT "A", verrà stampato il nuovo carattere.
Alcune brevi prove con questo procedimento dimostrano quanto questa funzionalità possa essere potente. Il programma che segue è un esempio di ridefinizione di un insieme di caratteri.

Byte 1
Byte 2
Byte 3
Byte 4
Byte 5
Byte 6
Byte 7
Byte 8

Figura C-1 - Quantità di memoria per carattere

Byte N°	Binary	Hex.	Decimal
1	00110000 =	30 =	48
2	00110000 =	30 =	48
3	11111000 =	F8 =	248
4	00011100 =	1C =	28
5	00001110 =	0E =	14
6	00000111 =	07 =	07
7	00000011 =	03 =	03
8	00000011 =	03 =	03

Figura C-2 - Ridefinizione di un carattere

Programma di esempio :

```
10 !
20 ! PROGRAMMA PER DIMOSTRARE
30 ! LA DEFINIZIONE DI UN SET
40 ! DI CARATTERI ALTERNATIVI
50 !
60 ! IL PROGRAMMA RIDEFINISCE I
70 ! CARATTERI A,B,C,D,E,F,G,H
80 !
90 CHRAS=&2F4!PUNTATORE AL SET DI CARATTERI
100 OPTION CHR1!ALLOCAZIONE DELL'AREA DI MEMORIA PER IL SET
110 ADDR%=VARPTR(CHR1)!   CALCOLO DELL'INDIRIZZO INIZIALE
120 PAGENO%=(ADDR%/256)AND&FF!CALCOLO DELLA PAGINA DI MEMORIA
130 !
140 MOVE 57344,ADDR%,1024!SPOSTA IL SET IN BASSO NELLA MEMORIA
150 !
160 OFFSET=33*8!OFFSET AD "A"
170 FOR I=0 TO 63!CARICA I NUOVI CARATTERI
180 READ C
190 POKE ADDR%+OFFSET+I,C!E INSERISCILO
200 NEXT I
210 !
220 !OGNI DATA CONTIENE UN CARATTERE
230 !
```

```
240 DATA &07,&0F,&1F,&3F,&7F,&FF,&FF,&FF
250 DATA &E0,&F0,&F8,&FC,&FE,&FF,&FF,&FF
260 DATA &FF,&FF,&FF,&7F,&3F,&1F,&0F,&07
270 DATA &FF,&FF,&FF,&FE,&FC,&F8,&F0,&E0
280 DATA &00,&00,&00,&EF,&7F,&FF,&FF,&FF
290 DATA &00,&00,&00,&FC,&FE,&FF,&FF,&FF
300 DATA &FF,&FF,&FF,&7F,&3F,&00,&00,&00
310 DATA &FF,&FF,&FF,&FE,&FC,&00,&00,&00
320 !
330 POKE CHBAS,PAGENO% !SCEGLI IL NUOVO SET DI CARATTERI!
340 !
350 POKE &2F0,1   ! SPEGNI IL CURSORE
360 SETCOLOR 6,2,6
370 X=20
380 FOR Y=10 TO 20
390 WAIT &D40B,&FF,110
400 CLS
410 PRINT AT(X,Y+1);"CD"
420 FOR W=1 TO 30:NEXT W
430 NEXT Y
440 CLS
450 PRINT AT(X,22);"GH"
460 SOUND 0,79,10,8,4
470 FOR W=1 TO 80: NEXT W
480 FOR Y=20 TO 10 STEP -1
490 WAIT &D40B,&FF,110
500 CLS
510 PRINT AT(X,Y+1);"CD"
520 FOR W=1 TO 30:NEXT W
530 NEXT Y
540 GOTO 380
```

Appendice D
Unita' di Ingresso – Uscita

GENERALITA'

La tastiera, l'unità dischi, il registratore e il modem sono mezzi da cui il calcolatore ottiene informazioni: queste vengono normalmente riferite come unità di ingresso o "input".
Inoltre, l'Home Computer ATARI fornisce informazioni scrivendole sullo schermo televisivo, su cassette, su stampante o su dischetti, che sono le unità di uscita o "output".

DEFINIZIONE DELLE UNITA'

Le unità di input-output ATARI hanno i seguenti codici di identificazione :

K:- Keyboard (Tastiera)
Unità di solo input. La tastiera permette al calcolatore di ottenere informazioni direttamente dai tasti della console.

P:- Line Printer (Stampante)
Unità di solo output. La stampante stampa caratteri ATASCII, una riga alla volta.

C:- Program Recorder (Registratore)
Unità di input/output. Il registratore è un'unità di lettura/scrittura che può essere usata in un modo o nell'altro, ma mai in tutti e due simultaneamente. La cassetta ha due tracce per registrazione del suono e del programma.
La traccia audio non può essere registrata dal Calcolatore ATARI, ma può essere riprodotta attraverso l'altoparlante della televisione.

D1:,-D2:,D3:,D4: Disk Drives (Unità a Dischi)

Unità di input e output. Il calcolatore ATARI può usare quattro Unità a Dischi, Atari Modello 810. Se non viene specificato nessun numero, il valore standard assunto è di 1.

E:- Screen Editor (Video)

Unità di input e output. Questa unità utilizza la tastiera e lo schermo televisivo (vedere S: Monitor TV) per simulare un terminale video. Scrivendo su tale unità, i dati appariranno su video iniziando dalla posizione attuale del cursore.

Leggendo da tale unità viene attivato il processo di screen-editing (modifica da video) e l'utente può introdurre ed editare dati. Ogni volta che si preme il tasto RETURN, l'intera riga viene interpretata come un record a se stante che l'unità centrale di input-output (CIO) trasferisce al programma utente.

Consultare il Manuale "Atari Home Computer - Note Tecniche di Riferimento" per una spiegazione dettagliata di CIO.

S:- Monitor TV (Monitor televisivo)

Unità di input/output. Questa unità permetterà di leggere caratteri dallo schermo e scrivere caratteri su di esso, usando il cursore come meccanismo di indirizzamento dello schermo. Vengono supportate operazioni grafiche e di testo.

R:- Interface RS-232 (Interfaccia)

Il modulo di interfaccia, Atari Modello 850 permette la connessione dei calcolatori Atari a unità compatibili con lo standard RS-232, come stampanti, terminali e plotters.

Appendice E
Posizioni di Memoria

GENERALITA'

Le posizioni di memoria sono espresse in esadecimale, con gli equivalenti decimali tra parentesi. Per ulteriori informazioni, vedere il Manuale "Atari Home Computer System - Note Tecniche di Riferimento".

MAPPA DELLA MEMORIA

Il microprocessore 6502 è diviso in quattro principali aree di memoria : area dedicata alla RAM, area dedicata alla cartuccia, area di residenza per i circuiti di input-output (I/O) e l'area in cui risiede il sistema operativo. Le aree di memoria ed i loro relativi limiti di indirizzamento sono elencati qui di seguito.

- Area RAM (minimo richiesto per operabilità)	0000-1FFF (0-8191)
- Area di ampliamento RAM	2000-7FFF (8192-32767)
- Cartuccia B (a sinistra) o RAM 8K	8000-9FFF (32768-40959)
- Cartuccia A (a destra) o RAM 8K	A000-BFFF (40960-49151)
- Non utilizzata	C000-CFFF (49152-53247)
- Circuiti di I/O	D000-D7FF (53248-55295)
- Pacchetto virgola mobile O.S.	D800-DFFF (55296-57343)
- ROM del sistema operativo residente	E000-FFFF (57344-65535)

AREA RAM

L'area dedicata alla RAM, condivisa tra il sistema operativo e il programma sotto controllo, è divisa nelle seguenti cinque regioni. Vedere Tabella E-1 per alcuni utili indirizzi del data base del S.O.

1 - Pagina 0; Regione modo indirizzo del Microprocessore 6502: da 0000 a 00FF (0-255) assegnata come segue :

- da 0000 a 007F (0-127) : Sistema Operativo
- da 0080 a 00FF (128-255) : Applicazione dell'utente
- da 00D4 a 00FF (212-255) : Package virgola mobile, se usato

2 - Pagina 1; Regione per il controllo del Hardware 6502 : da 0100 a 01FF (256-511).

NOTA

All'accensione o dopo un SYSTEM RESET, la pila di memoria punta all'indirizzo 01FF (511) e poi la pila torna indietro verso l'indirizzo 0100 (256). Se si verifica un superamento della memoria, la pila si avvolge intorno da 0100 a 01FF.

3 - Pagine 2-4; Regione assegnata alla base Dati del S.O. (variabili di lavoro, tabelle, buffers dati): da 0200 a 047F (512-1151).

4 - Pagine 7-XX; Regione assegnata al contenimento dell'area di caricamento dell'utente 0700 (1792) sino all'inizio dell'area libera della RAM, dove XX è una funzione del modo grafico dello schermo e del valore di RAM installata.

NOTA

Una volta completata l'inizializzazione del dischetto, la variabile del data base punta alla successiva locazione disponibile sopra al software caricato. Se al momento dell'inizializzazione del dischetto non é stato caricato alcun software, la variabile del data base punta alla locazione 0700.

5 - Regione da pagina XX alla fine della RAM per Lista e Dati di visualizzazione dello schermo. Il puntatore del data base contiene l'indirizzo dell'ultima posizione disponibile sotto l'area dello schermo.

AREA CARTUCCE

La cartuccia B è quella a destra sull'Home Computer, Atari Modello 800, mentre la cartuccia A è quella a sinistra e la sola presente sull'Home Computer 400.

- Cartuccia B : da 8000 a 9FFF (32768-40959)
- Cartuccia A : da A000 a BFFF (40960-49151) per cartucce da 8K
  da 8000 a BFFF (32768-49151) per cartucce da 16K (opzionali)

NOTA

Sull'Home Computer, Atari Modello 800, se un modulo RAM inserito nell'ultima fessura si sovrappone a qualcuno dei suddetti indirizzi di cartuccia, la cartuccia installata disabilita il modulo RAM ad incrementi di 8K.

CIRCUITI DI I/O

Il microprocessore 6502 esegue operazioni di input/output indirizzando i seguenti circuiti di supporto esterno come memoria:

- GTIA/CTIA da D000 a D01F (53248-53279)
- POKEY da D200 a D21F (53760-53791)
- PIA da D300 a D31F (54016-54047)
- ANTIC da D400 a D41F (54272-54303)

Alcuni registri dei suddetti circuiti sono di lettura/scrittura; altri sono solo di scrittura o solo di lettura. La Tabella E-2 elenca i registri ed i loro indirizzi suddivisi per circuito. Per ulteriori informazioni vedere il Manuale "Atari Home Computer System - Note Tecniche di Riferimento".

ROM DEL S.O. RESIDENTE

L'area da D800 a FFFF (55296-65535) contiene permanentemente il Sistema Operativo ed il package della virgola mobile:

- Package virgola mobile: da D800 a DFFF (55296-57343)
- ROM del Sistema Operativo: da E000 a FFFF (57344-65535)

Il Sistema Operativo contiene molti punti di entrata vettoriali, tutti fissi, alla fine della ROM e nella RAM.

Il package virgola mobile non è vettoriale, ma tutti i punti di entrata sono fissi. I listings dei vettori e dei punti di entrata fissi della ROM sono elencati nell'Appendice del Manuale "Atari Home Computer System - Note Tecniche di Riferimento".

☐ Tabella E-1 INDIRIZZI UTILI DEL DATA BASE DEL S.O.

• CONFIGURAZIONE DELLA MEMORIA

INDIRIZZO Esadec.	Decim.	Nome	Dim. Byte	Funzione
000E	14	APPMHI	2	Limite inferiore dello schermo della memoria libera per l'utente
006A	106	RAMTOP	1	Parte alta del gestore video dell'indirizzo RAM (MSB)
02E4	740	RAMSIZ	1	Parte alta di indirizzo RAM (MSB)
02E5	741	MEMTOP	2	Indirizzo alto della memoria libera per l'utente
02E7	743	MEMLO	2	Indirizzo basso della memoria libera per l'utente

• SCHERMO GRAFICI/TESTO

Margini di schermo (modi di testo; area di testo)

0052	82	LMARGN	1	Margine sinistro dello schermo (0-39; standard 2)
0053	83	RMARGN	1	Margine destro dello schermo (0-39; standard 39)

Controllo cursore

0054	84	ROWCRS	1	Riga attuale del cursore
0055	85	COLCRS	2	Colonna attuale del cursore
005A	90	OLDROW	1	Riga precedente del cursore
005B	91	OLDCOL	2	Colonna precedente del cursore
0290	656	TXTROW	1	Riga attuale del cursore nell'area del testo
0291	657	TXTCOL	2	Colonna attuale del cursore nell'area di testo
02F0	752	CRSINH	1	Indicatore per inibire la visualizzazione del cursore (0 = cursore acceso; 1 = cursore spento)

Controllo colore

INDIRIZZO Esadec.	Decim.	Nome	Dim. Byte	Funzione
02C0	704	PCOLR0	4	Colore-luminanza del giocatore e missile 0
02C1	705	PCOLR1	4	Colore-luminanza del giocatore e missile 1
02C2	706	PCOLR2	4	Colore-luminanza del giocatore e missile 2
02C3	707	PCOLR3	4	Colore-luminanza del giocatore e missile 3
02C4	708	COLOR0	5	Colore-luminanza del giocatore e missile 0
02C5	709	COLOR1	5	Colore-luminanza del giocatore e missile 1
02C6	710	COLOR2	5	Colore-luminanza del giocatore e missile 2
02C7	711	COLOR3	5	Colore-luminanza del giocatore e missile 3
02C8	712	COLOR4	5	Colore-luminanza dello sfondo

Modo "Attract" (Attrazione)

004D	77	ATRACT	1	Indicatore e temporizzatore modo "ATTRACT" (Valore 128 = acceso; si accende ogni 9 minuti)

Tabulazione

02A3	675	TABMAP	15	Mappa dei bits di arresto tabulazione (standard: 7,15,23, ecc. fino a 119)

Memoria dello schermo

0058	88	SAVMSC	2	Angolo superiore sinistro dello schermo

Memoria dello schermo suddiviso

0294	660	TXTMSC	2	Angolo superiore sinistro dell'area di testo

● FUNZIONI DRAW/FILL

02FD	765	FILDAT	1	Dati di riempimento per il comando grafico FILL

Conversione interna del codice caratteri

02FA	762	ATACHR	1	Contiene l'ultimo punto disegnato o carattere ATASCII

Caratteri di controllo visualizzazione

INDIRIZZO Esadec.	Decim.	Nome	Dim. Byte	Funzione
02FE	766	DSPFLG	1	Indicatore dei caratteri di controllo visualizzazione (1 = caratteri di controllo visualizzazione).

- TASTIERA

Lettura dei tasti

02FC	764	CH	1	Contiene il valore dell'ultimo carattere della tastiera in FIFO o $FF se FIFO è vuoto

Funzioni speciali

0011	17	BRKKEY	1	Indicatore del tasto BREAK (normalmente diverso da zero; forzato a zero da BREAK)
02B6	694	INVFLG	1	Indicatore inversione video (normalmente = 0, abilitato dal tasto ■).
02BE	702	SHFLOK	1	Indicatore controllo per i tasti shift/control e lock ($00 = nessun blocco, normale; $40 = blocco maiuscole; $80 = blocco controllo) Posizionato su $40 all'accensione e al SYSTEM RESET; riposizionato da : CAPS LOWR, CAPS LOWR SHIFT, o CAPS LOWR CTRL.
02FF	767	SSFLAG	1	Indicatore di start/stop (normale = 0, abilitato da CTRL 1)

- ROUTINE DI I/O CENTRALE

Blocco per controllo I/O

0340-034F (832-847) IOCB	16	Blocco controllo I/O 0
0350-035F (848-863) IOCB	16	Blocco controllo I/O 1
0360-036F (864-879) IOCB	16	Blocco controllo I/O 2
0370-037F (880-895) IOCB	16	Blocco controllo I/O 3
0380-038F (896-911) IOCB	16	Blocco controllo I/O 4
0390-039F (912-927) IOCB	16	Blocco controllo I/O 5
03A0-03AF (928-943) IOCB	16	Blocco controllo I/O 6
03B0-03BF (944-959) IOCB	16	Blocco controllo I/O 7

INDIRIZZO Esadec.	Decim.	Nome	Dim. Byte	Funzione
0340	832	ICHID	1	Identificatore gestore unità (vedere Sez.5; inizializzato a $FF all'accensione e al SYSTEM RESET)
0341	833	ICDNO	1	Numero unità
0342	834	ICCMD	1	Byte del comando
0343	835	ICSTA	1	Stato
0344	836	KBAL/ICBAH	2	Indirizzo del buffer
0346	838	ICPTL/ICPTH	2	Vettore PUT BYTE (indica IOCB non aperti del CIO all'accensione e al SYSTEM RESET).
0348	840	ICBLL/ICBLH	2	Contatore lunghezza buffer/byte
034A	842	ICAX1/ICAX2	2	Informazioni ausiliarie
034C	844	ICAX3/ICAX6	4	Bytes di riserva per uso del gestore delle unità

Pagina zero IOCB

0020	32	ZIOCB	16	Pagina zero IOCB (Solo i primi 12 bytes (IOCB) vengono spostati dalla funzione CIO.
0020	32	ICHIDZ	1	Numero indice del gestore di unità (posiziona to a $FF su CLOSE)
0021	33	ICDNOZ	1	Numero dell'unità
0022	34	ICCOMZ	1	Byte di comando
0023	35	ICSTAZ	1	Byte di stato
0024	36	ICBALZ,ICBALH	2	Indirizzo del byte
0026	38	ICPTLZ,ICPTHZ	2	Vettore PUT BYTE (indica IOCB non aperti di CIO nella CLOSE)
0028	40	ICBLLZ,ICBLHZ	2	Contatore lunghezza buffer/byte
002A	42	ICAX1Z,ICAX2Z	2	Informazioni ausiliarie
0002C	44	ICSPRZ (ICIDNO,ICOCHR)	4	Variabili di lavoro CIO CIDNO = ICSPRZ + 2; ICOCHR = ICSPRZ + 3

- **STATO DELL'UNITA'**

02EA	746	DVSTAT	4	Stato dell'unità

- **TABELLA DELL'UNITA'**

031A	749	HATABS	38	Tabella del gestore unità

• ROUTINE PER I/O SERIALE (SIO)

Blocco di controllo unità

INDIRIZZO Esadec.	Decim.	Nome	Dim. Byte	Funzione
0300-030B	(768-779)	DCB	12	Blocco di controllo unità
0300	768	DDEVIC	1	Identificatore del bus unità
0301	769	DUNIT	1	Numero di unità
0302	770	DCOMND	1	Comando unità
0303	771	DSTATS	1	Stato dell'unità
0304	772	DBUFLO,DBUFHI	2	Indirizzo del buffer del gestore di unità
0306	774	DTIMLO	1	Fine del tempo unità disponibile
0308	776	DBYTLO,DBYTHI	2	Contatore lunghezza buffer/byte
030A	778	DAUX1,DAUX2	2	Informazioni ausiliarie

• CONTROLLO DEL SUONO

0041	65	SOUNDR	1	Indicatore I/O silenzioso/rumoroso (0 = silenzioso)

• GOVERNI ATARI

Governi a cloche

0278	632	STICK0-STICK3	4	Bocchettone posizione cloche
0284	644	STRIG0-STRIG3	4	Bocchettone grilletto cloche

Governi a manopola

0270	624	PADDL0-PADDL7	8	Bocchettone posizione manopola
027C	636	PTRIG0-PTRIG7	8	Bocchettone grilletto manopola

Penna luminosa

0234	564	LPENH	1	Codice posizione orizzontale penna luminosa
0235	565	LPENV	1	Codice posizione verticale penna luminosa
0278	632	STICK0-STICK3	4	Bocchettone pulsante penna luminosa

● PACCHETTO VIRGOLA MOBILE

INDIRIZZO Esadec.	Decim.	Nome	Dim. Byte	Funzione
00D4	212	FR0	6	Registro virgola mobile 0
00E0	224	FR1	6	Registro virgola mobile 1
00F2	242	CIX	1	Index caratteri
00F3	243	INBUFF	1	Puntatore del buffer del testo di input
00FB	251	DEGFLG/RADFLG	1	Indicatore/gradi radianti (0 = DEGFLG; 6 = gradi; DEGFLG = 0)
00FC	252	FLPTR	2	Puntatore al numero virgola mobile
0580	1408	LBUFF	96	Buffer testo

● ACCENSIONE E SYSTEM RESET

Caricamento Dischetto/Cassetta

0002	2	CASINI	2	Vettore inizializzazione caricamento cassetta
000C	12	DOSINI	2	Vettore inizializzazione caricamento dischetto

Controllo ambiente

0008	8	WARMST	1	Indicatore avvio a caldo (= 0 alla accensione; $FF su SYSTEM RESET)
000A	10	DOSVEC	2	Vettore controllo non cartuccia

● INTERRUZIONI

0010	16	POKMSK	1	Maschera interruzione POKEY
0042	66	CRITIC	1	Indicatore sezione codice critica (non zero = codice esecutivo è critico)

Clock in tempo reale

0012	18	RTCLOK	3	Contatore del tempo reale (1/60 sec.) (+ 0 = MSB; + 1 = NSB; + 2 = LSB)

Temporizzatori sistema VBLANK

0218	536	CDTMV1	2	Valore del temporizzatore di sistema 1
021A	538	CDTMV2	2	Valore del temporizzatore di sistema 2
021C	540	CDTMV3	2	Valore del temporizzatore di sistema 3
021E	542	CDTMV4	2	Valore del temporizzatore di sistema 4
0020	544	CDTMV5	2	Valore del temporizzatore di sistema 5

INDIRIZZO Esadec.	Decim.	Nome	Dim. Byte	Funzione
0226	550	CDTMA1	2	Indirizzo di salto temporizzat. di sistema 1
0228	552	CDTMA2	2	Indirizzo di salto temporizzat. di sistema 2
022A	554	CDTMF3	2	Indicatore temporizzat. di sistema 3
022C	556	CDTMF4	1	Indicatore temporizzat. di sistema 4
022E	558	CDTMF5	2	Indicatore temporizzat. di sistema 5

Vettori di interruzione NMI

0200	512	VDSLST	2	Vettore interruzione lista (non usato dal S.O.)
0222	546	VVBLKI	2	Vettore VBLANK immediato
0224	548	VVBLKD	2	Vettore VBLANK differito

Vettori di interruzione IRQ

0202	514	VPRCED	2	Segnale prosecuzione bus I/O seriale
0204	516	VINTER	2	Segnale interruzione bus I/O seriale
0206	518	VBREAK	2	Vettore istruzione BREAK
0208	520	VKEYBD	2	Vettore interruzione tastiera
020A	522	VSERIN	2	Pronto a ricevere dati del bus seriale di I/O
020C	524	VSEROR	2	Pronto a trasmettere dati del bus seriale di I/O
020E	526	VSEROC	2	Trasmissione completata dal bus seriale di I/O
0210	528	VTIMR1	2	Vettore temporizzatore POKEY (non usato dal S.O.)
0212	530	VTIMR2	2	Vettore temporizzatore POKEY (non usato dal S.O.)
0214	532	VTIMR4	2	Vettore temporizzatore POKEY (non usato dal S.O.)
0216	534	VIMIRQ	2	Vettore IRQ generale

Aggiornamenti dei registri hardware

0230	560	SDLSTL	1	Indirizzo lista di visualizzazione schermo
0231	561	SDLSTH	1	Indirizzo lista di visualizzazione schermo
02C0	704	PCOLRx	4	Registro colori
02C4	708	PCOLORx	5	Registro colori
02F3	755	CHACT	1	Controllo caratteri
02F4	756	CHBAS	1	Registro base degli indirizzi carattere ($E0 = set di numeri maiuscoli; $E2 = set di caratteri grafici speciali minuscoli; standard = $E0)

AREE UTENTE

Le seguenti aree sono disponibili per l'utente in un ambiente non nidificato.

0080	128		128	
0480	1152		640	

TABELLA E-2 INDIRIZZI HARDWARE

• PIASTRINA ANTIC

INDIRIZZO Esad.	Decim.	Nome Reg.	Funzione	S.O. Dec.	Falso Dec.	Nome
0400	54272	DMACTL	Controllo (WRITE) accesso diretto alla memoria (DMA)	22F	559	SDMCTL
0401	54273	CHACTL	Controllo carattere (WRITE)	2F3	755	CHART
0402	54274	DLISTL	Byte basso del puntatore lista visualizzazione	230	560	SDLSTL
0403	54275	DLISTH	Byte alto del puntatore lista visualizzazione (WRITE).	231	561	SDLSTH
0404	54276	HSCROL	Scorrimento orizzontale (WRITE)			
0405	54277	VSCROL	Scorrimento verticale			
0407	54279	PMBASE	Indirizzo base giocatore e missile(WRITE)			
0409	54281	CHBASE	Indirizzo base caratteri (WRITE)	2F4	756	CHBAS
040A	54282	WSYNC	Attesa per sincronismo orizzontale (WRITE)			
040B	54283	VCOUNT	Contatore riga verticale (READ)			
040E	54286	NMIEN	Abilitazione interruzione non mascherabile (NMI) (WRITE)			
040F	54287	NMIRES	Reset NMIST (WRITE)			
040F	54287	NMIST	Stato NMI (READ)			
0410 a 04FF	(54288 a 54527)		Ripete gli indirizzi di ANTIC da 0400 a 040F			

• PIASTRINA CTIA/GTIA

Controllo posizione orizzontale (WRITE)

D000	53248	HPOSP0	Posizione orizzontale giocatore 0			
D001	53249	HPOSP1	Posizione orizzontale giocatore 1			
D002	53250	HPOSP2	Posizione orizzontale giocatore 2			
D003	53251	HPOSP3	Posizione orizzontale giocatore 3			
D004	53252	HPOSM0	Posizione orizzontale missile 0			
D005	53253	HPOSM1	Posizione orizzonrale missile 1			
D006	53254	HPOSM2	Posizione orizzontale missile 2			
D007	53255	HPOSM3	Posizione orizzontale missile 3			

Controllo collisioni (READ)

INDIRIZZO Esad.	Decim.	Nome Reg.	Funzione	S.O. Dec.	Falso Dec.	Nome
D000	53248	M0PF	Tra missile 0 e campo di gioco			
D001	53249	M1PF	Tra missile 1 e campo di gioco			
D002	53250	M2PF	Tra missile 2 e campo di gioco			
D003	53251	M3PF	Tra missile 3 e campo di gioco			
D004	53252	P0PF	Tra giocatore 0 e campo di gioco			
D005	53253	P1PF	Tra giocatore 1 e campo di gioco			
D006	53254	P2PF	Tra giocatore 2 e campo di gioco			
D007	53255	P3PF	Tra giocatore 3 e campo di gioco			
D008	53256	M0PL	Tra missile 0 e giocatore			
D009	53257	M1PL	Tra missile 1 e giocatore			
D00A	53258	M2PL	Tra missile 2 e giocatore			
D00B	53259	M3PL	Tra missile 3 e giocatore			
D00C	53260	P0PL	Tra giocatore 0 e giocatore			
D00D	53261	P1PL	Tra giocatore 1 e giocatore			
D00E	53262	P2PL	Tra giocatore 2 e giocatore			
D00F	53263	P3PL	Tra giocatore 3 e giocatore			

Azzeramento collisione (WRITE)

D01E	53278	HITCLR	Azzeramento collisione			

Controllo dimensione (WRITE)

Nota : 0 = dimensione normale, 1 = doppia, 3 = quadrupla

D008	53256	SIZEP0	Dimensione del giocatore 0			
D009	53257	SIZEP1	Dimensione del giocatore 1			
D00A	53258	SIZEP2	Dimensione del giocatore 2			
D00B	53259	SIZEP3	Dimensione del giocatore 3			
D00C	53260	SIZEM	Dimensione di tutti i missili			

Registri grafici (WRITE)

D00D	53261	GRAFP0	Grafici per giocatore 0			
D00E	53262	GRAFP1	Grafici per giocatore 1			
D00F	53263	GRAFP2	Grafici per giocatore 2			
D010	53264	GRAFP3	Grafici per giocatore 3			
D01	53265	GRAFM	Grafici per tutti i missili			

Grilletto del comando a cloche (READ)

INDIRIZZO Esad.	Decim.	Nome Reg.	Funzione	S.O. Dec.	Falso Dec.	Nome
D010	53264	TRIG0	Leggi grilletto del comando cloche 0	284	644	STRIG0
D011	53265	TRIG1	Leggi grilletto del comando cloche 1	285	645	STRIG1
D012	53266	TRIG2	Leggi grilletto del comando cloche 2	286	646	STRIG2
D013	53267	TRIG3	Leggi grilletto del governo cloche 3	287	647	STRIG3

Controllo luminanza-colore (WRITE)

D012	53266	COLPM0	Colore-luminanza giocatore e missile 0	2C0	704	COLR0
D013	53267	COLPM1	Colore-luminanza giocatore e missile 1	2C1	705	PCOLR1
D014	53268	COLPM2	Colore-luminanza giocatore e missile 2	2C2	706	PCOLR2
D015	53269	COLPM3	Colore-luminanza giocatore e missile ´	2C3	707	PCOLR3
D016	53270	COLPF0	Colore-luminanza campo di gioco 0	2C4	708	COLOR0
D017	53271	COLPF1	Colore-luminanza campo di gioco 1	2C5	709	COLOR1
D018	53272	COLPF2	Colore-luminanza campo di gioco 2	2C6	710	COLOR2
D019	53273	COLPF3	Colore-luminanza campo di gioco 3	2C7	711	COLOR3
D01A	53274	COLBK	Colore-luminanza sfondo	2C8	712	COLOR4

Controllo priorità (WRITE)

D01B	53275	PRIOR	Selezione priorità	26F	623	GPRIOR

Controllo grafici (WRITE)

D01D	53277	GRACTL	Controllo grafici

• FUNZIONI MISCELLANEE DI I/O

Sistemi PAL/NTSC

D014	53268	PAL	Leggere bits PAL/NTSC

Interruttore di console (posizionati su 8 durante VBLANK)

D01F	53279	CONSOL	Interruttore bocchettone console a scrittura
D01F	53279	CONSOL	Interruttore bocchettone console a lettura

● PIASTRINA POKEY

Audio (WRITE)

INDIRIZZO Esad.	Decim.	Nome Reg.	Funzione	S.O Dec.	Falso Dec.	Nome
D200	53760	AUDF1	Frequenza canale audio 1			
D201	53761	AUDC1	Controllo canale audio 1			
D202	53762	AUDF2	Frequenza canale audio 2			
D203	53763	AUDC2	Controllo canale audio 2			
D204	53764	AUDF3	Frequenza canale audio 3			
D205	53765	AUDC3	Controllo canale audio 3			
D206	53766	AUDF4	Frequenza canale audio 4			
D207	53767	AUDC4	Controllo canale audio 4			
D208	53768	AUDCTL	Controllo audio			

Temporizzatore di avvio

D209	53769	STIMER	Resetta i separatori frequenza-audio ai valori AUDF			

Scansione potenziometri (Governi a manopola)

D200	53760	POT0	Leggere il pot. 0	270	624	PADDL0
D201	53761	POT1	Leggere il pot. 1	271	625	PADDL1
D202	53762	POT2	Leggere il pot. 2	272	626	PADDL2
D203	53763	POT3	Leggere il pot. 3	273	627	PADDL3
D204	53764	POT4	Leggere il pot. 4	274	628	PADDL4
D205	53765	POT5	Leggere il pot. 5	275	629	PADDL5
D206	53766	POT6	Leggere il pot. 6	276	630	PADDL6
D207	53767	POT7	Leggere il pot. 7	277	631	PADDL7
D208	53768	ALLPOT	Leggere lo stato del bocchettone-pot.-riga 8			
D208	53771	POTGO	Avviare sequenza scansione potenziometri (scritta durante VBLANK)			

Controllo e Scansione Tastiera (READ)

D209	53769	KBCODE	Codice tastiera	2FC	764	CH

Generatore numeri casuali (READ)

D20A	53770	RANDOM	Generatore numeri casuali			

Bocchettone seriale

INDIRIZZO Esad.	Decim.	Nome Reg.	Funzione	S.O. Dec.	Falso Dec.	Nome
D20A	53770	SKRES	Reset SKSTAT (WRITE)			
D20D	53773	SERIN	Input del bocchettone seriale (READ)			
D20D	53773	SEROUT	Output del bocchettone seriale (WRITE)			
D20F	53775	SKCTLS	Controllo 4 tastiere del bocchettone seriale (WRITE)	232	562	SSKCTL
D20F	53775	SKSTAT	Registro di stato delle 4 tastiere del bocchettone seriale (READ)			

Interruzione IRQ

D20E	532774	IRQEN	Abilitazione interruzione IRQ (WRITE)	10	16	POKMSK
D20E	532775	IRQST	Stato interruzione IRQ (READ)			
D210 a D2FF	(53776 a 54015)		Ripetere da D200 a D20F (53760-53775)			

● PIASTRINA PIA

Registri Lettura/Scrittura Governi a cloche

D300	54016	PORTA	Scrive o legge dati dai Jacks 1 e 2 del governo se bit 2 di PACTL = 1 Scrive sul registro controllo direzione se il bit 2 di PACTL = 0	278 279	632 633	STICK0 STICK1
D301	54017	PORTB	Scrive o legge dati dai jacks 3 e 4 del governo se il bit 2 di PBCTL = 1 Scrive su registro di controllo direzione se il bit 2 = PBCTL = 0	27A 27B	634 635	STICK2 STICK3
D302	54018	PACTL	Controllo bocchettone A (posizionato su $3C dal codice IRQ)			
D303	54019	PBCTL	Controllo bocchettone B (posizionato su $3C dal codice IRQ)			
D304 a D3FF	(54020 a 54271)		Ripetere D300-D303 (54016-54019)			

Note

Appendice F
Conversione di Programmi

NOTE SUI DIRITTI DI AUTORE

I programmi dei calcolatori sono protetti dalle Leggi relative ai Diritti di Autore (copyright). Il regolare possessore di una copia di un programma protetto da diritti d'autore, può di regola adattare tale programma per farlo girare sulla sua macchina. Comunque, ci sono dei limiti a questo diritto e particolari personalizzazioni non possono essere trasferite a terzi senza l'autorizzazione scritta del detentore del diritto di autore.

CONVERSIONE DI PROGRAMMI
IN MICROSOFT BASIC II ATARI

Il COMMODORE PET* BASIC, l'APPLE** APPLESOFT** BASIC e il RADIO SHACK*** LEVEL II BASIC sono stati scritti tutti dalla Microsoft. L'approccio e le sintassi generali di questi linguaggi BASIC sono stati mantenuti compatibili nei limiti del possibile per permettere ai programmi ed ai programmatori di spostarsi facilmente da una macchina all'altra. Questa appendice elenca le differenze esistenti tra i linguaggi ed evidenzia eventuali modifiche da apportare qualora si desideri eseguire una conversione nel Microsoft BASIC II della Atari.

Il Microsoft ha diviso il BASIC originale in vari livelli differenti: 4K, 8K, ampliato e completo.

Inizialmente, ogni livello successivo rappresentava un ampliamento al precedente e perciò richiedeva più memoria. Quando veniva commissionato lo sviluppo del BASIC, il livello terminale delle funzionalità veniva stabilito in base ai limiti di memoria della macchina a cui era destinato. I vari livelli di incompatibilità furono quindi determinati non solo dal tipo di macchina sulla quale avrebbero funzionato, ma anche dal livello di partenza. Infatti, il PET BASIC e l'APPLE APPLESOFT BASIC hanno una base di sviluppo centrata sul livello 8K, mentre il RADIO SHACK LEVEL II ed il Microsoft BASIC II della Atari sono basati sull'ottenimento delle prestazioni massime del linguaggio. Fortunatamente, ciò facilita operazioni di conversione nel Microsoft BASIC II della Atari.

Le differenze principali tra un linguaggio a 8K ed il Basic completo sono le seguenti :

1 - Nel BASIC 8K non viene supportata la doppia precisione. I numeri interi vengono convertiti in semplice precisione prima che sia eseguita qualsiasi operazione matematica, così il loro vantaggio è la poca memoria occupata,non la velocità.

2 - PRINT USING non è disponibile, perciò l'utente è obbligato a formattare i propri numeri in BASIC 8K.

3 - Gli statements avanzati:IF...THEN...ELSE , DEFINT, DEFSNG, DEFDBL, DEFSTR, TRON, TROFF, RESUME e LINE INPUT non sono supportati nel BASIC 8K.

4 - Le funzioni INSTR e STRING$ non sono supportate nel BASIC 8K.

5 - Nel BASIC 8K le matrici possono essere solo a dimensione singola.

6 - Le funzioni definite dall'utente possono avere un solo argomento.

Le aree che presentano maggior difficoltà di conversione sono quelle che interessano le funzioni dipendenti dalla macchina come i grafici e l'uso del linguaggio macchina.

* PET è un marchio registrato della Commodore Business Machine Inc.
**APPLE e APPLESOFT sono marcvhi registrati della APPLE COMPUTER.
***RADIO SHACK è un marchio registrato della TANDY CORPORATION.

Appendice G
Conversione da Commodore (PET) Versione 4.0 a Microsoft BASIC II

GENERALITA'

Le maggiori difficoltà riscontrabili nella conversione dal BASIC Commodore (PET), usato su calcolatori, Commodore PET, deriva dalle specifiche caratteristiche dell'hardware piuttosto che dal linguaggio BASIC dato che si tratta di una stretta implementazione del livello 8K.

Alcune considerazioni inerenti alla conversione sono :

DIFFERENZE

1 - Il set di caratteri Commodore PET è stato ampliato a 256 caratteri.
Questi caratteri sono caratteri realizzati a blocchi grafici. Per emulare questa funzione del Commodore PET, è necessario creare un set di caratteri nella RAM del Home Computer ATARI.

2 - Il PET BASIC della Commodore ha le seguenti costanti incorporate: TI$ (TIME$ per Calcolatori ATARI) e TI (TIME per calcolatori ATARI) ST per lo STATUS dell'ultima operazione di I/O e un simbolo pi (pi greca) per la costante pi greca.

3 - Le operazioni di I/O sul PET Commodore vengono eseguite con statements speciali che controllano il bus IEEE. Gli argomenti della OPEN sono completamente differenti dalle altre macchine e devono essere modificati completamente. Il formato esatto di inizio caratteri viene fatto specificando un numero di canale con gli statement PRINT e INPUT, che corrispondono a quelli del Microsoft BASIC II ATARI, perciò solo gli statements di controllo e OPEN devono essere riprogrammati.

4 - La dimensione dello schermo del PET Commodore è 40 colonne per 25 righe. Se i menu sono stati realizzati per questa configurazione, devono essere riprogrammati.

5 - I comandi PEEK e POKE sono molto influenzati dall'architettura della macchina. I programmi PET Commodore usano spesso PEEK e POKE per controllare il posizionamento del cursore perchè non c'e alcun modo diretto per modificarne la posizione. Ogni PEEK e POKE deve essere esaminato e riprogrammato.

6 - I programmi PET Commodore hanno spesso caratteri di controllo del cursore incorporati nelle stringhe di testo. Il Microsoft BASIC II Atari gestisce anche questa funzione, ma i codici carattere sono diversi e devono essere modificati.

7 - Il Pet Commodore chiama CLEAR CLR.

8 Ogni utilizzo del linguaggio macchina attraverso lo statement EXEC del PET Commodore deve essere attentamente esaminato poichè, sebbene il microprocessore sia lo stesso, la configurazione della memoria ed il modo di passare gli argomenti al BASIC e di riceverli da esso sono alquanto diversi.

9 - Dato che il PET Commodore non gestisce funzioni sonore o grafiche in senso stretto, non è possibile nessuna conversione in queste aree.

10- L'istruzione RND funziona in modo diverso; infatti RND, con un argomento positivo (generalmente 1),restituisce un numero compreso tra 0 e 1.

In linea generale, se viene generato un set di caratteri uguale a quello disponibile sul PET Commodore, non dovrebbere insorgere particolari difficoltà nella conversione di programmi che non fanno pesante uso del linguaggio macchina o di istruzioni tipo PEEK o POKE. Usare la seguente tabella per convertire un programma software sviluppato in versione 4.0 sul PET Commodore.

I comandi in BASIC universalmente accettati, come RUN, COUNT e POKE sono stati omessi. In questi casi infatti non è necessaria alcuna conversione.
Questa tabella può essere usata anche per eseguire funzioni su dischetto. Il BASIC 4.0 del PET Commodore è un linguaggio basato su dischetto che deve essere supportato dalle opzioni DOS dei Calcolatori ATARI.

Comando Commodore PET	Equivalente OPZIONE DOS dei Calcolatori ATARI	Microsoft BASIC ATARI
DIRECTORY	A RETURN DIRECTORY-SEARCH SPEC, LIST FILE? RETURN	
COPY	C RETURN COPY-FROM, TO? D1: Nome file, D2: nome file RETURN	
RENAME	E RETURN RENAME, GIVE OLD NAME, NEW D2: vecchio nome file, nuovo nome file RETURN	NAME

Comando Commodore PET	Equivalente OPZIONE DOS dei Calcolatori ATARI	Microsoft BASIC ATARI
SCRATCH	D RETURN DELETE FILESPEC D2: nome file RETURN TYPE "Y" TO DELETE nome file Y RETURN	KILL
HEADER	I RETURN WHICH DRIVE TO FORMAT?	

Controllare la logica del software che si vuole convertire per stabilire la direzione di questi comandi. Il loro uso va programmato sulla base dei risultati che si vogliono ottenere con applicazioni software-personalizzate.

Appendice H
Conversione da TRS Radio Shack a Microsoft BASIC II

Il BASIC Radio Shack è basato sul Microsoft BASIC completo, perciò i programmi convertiti faranno un uso molto migliore delle funzioni del Microsoft BASIC II Atari di quanto non succeda con i programmi APPLE e PET 'Commodore. Il Microsoft BASIC II Atari possiede funzioni aggiuntive come ad esempio la COMMON, poichè è stato scritto più recentemente e le limitazioni alla quantità di memoria disponibile per memorizzare il BASIC stesso non è così estesa sui calcolatori Radio Shack quanto lo sia sui calcolatori Atari. Il termine "Radio Shack" BASIC si riferisce al BASIC incorporato nei calcolatori Modello I e Modello III e viene chiamato BASIC di Livello II. Il BASIC gestito dal Modello II è molto simile ma non viene considerato in questa sede.

DIFFERENZE

1 - La dimensione del video del Radio Shack pone il più grande problema per la conversione di programmi TRS-80 BASIC poichè è di 16 righe per 64 colonne. I programmi che usano tutte le 64 colonne per Tabelle e menu devono essere modificati.

2 - Radio Shack gestisce una forma di grafica che permette visualizzazioni in bianco e nero di 128 x 48 pixels frammisti con caratteri. Gli statements per la manipolazione della grafica sono: CLS (azzerare lo schermo), SET (accendere un punto), RESET (spegnere un punto) e POINT (verificare il valore di un punto dello schermo).

3 - Radio Shack non memorizza la freccia verso l'alto nella posizione ASCII standard, perciò tale carattere deve essere tradotto quando programmi TRS-80 vengono utilizzati su calcolatori Atari.

4 - Il Radio Shack attua la stampa attraverso i comandi LPRINT e LLIST senza aprire (OPEN#...) l'unità. L'I/O su cassetta viene eseguito con PRINT o INPUT sui canali 1 e 2 (possono essere gestite due unità). Il formato dei files su cassetta, inoltre, è completamente differente.

5 - Richiami al linguaggio macchina vengono eseguiti con USR. Dato che i calcolatori Radio Shack utilizzano il microprocessore Z-80 invece del 6502, le routine in linguaggio macchina devono essere completamente riscritte.

6 - I comandi PEEK e POKE non possono essere convertiti direttamente. PEEK e POKE non vengono usati molto spesso sui calcolatori Radio Shack.

7 - La sintassi di posizionamento del cursore è un @ dopo PRINT nel BASIC Radio Shack e "AT" nel Microsoft BASIC II ATARI.

8 - I codici di errore restituiti dal comando ERR sono completamente differenti.

TRS-80	ATARI	DEFINIZIONE
CDBL(exp)	-------	Restituisce la rappresentazione in doppia precisione della espressione.
CINT(exp)	-------	Restituisce il numero intero più alto, non maggiore della espressione.
CLOAD	CLOAD LOAD "C"	Carica un programma BASIC da nastro.
CLOAD?	VERIFY"C:filespec"	Verifica un programma BASIC su nastro con uno in memoria.
CSNG(X)	Tronca automaticamente	Restituisce la rappresentazione in semplice precisione della espressione.
EDIT in	AUTO numero riga	Permette di editare un determinato numero di riga. Usa tasti di controllo del cursore.
FIX(x)	SGN(X)*INT(ABS(X))	Tronca tutte le cifre a destra del punto decimale.
INPUT # -1	OPEN # 5, "C:" INPUT INPUT # 5	INPUT legge dati da cassetta.
LLIST	LIST "P:" mm-nn	Lista programmi sulla stampante
LPRINT	OPEN # 4, "P:" OUTPUT PRINT # 4, "TEST"	Stampa una riga sulla stampante
PRINT MEM	PRINT FRE(0)	Stampa la quantità di memoria disponibile.
POINT (X,Y)	OPEN # 5, "D:" INPUT o GET # iocb, AT(s.b) INPUT # 5, AT(s.b) o PUT # iocb, AT(s.b) (s = settore/b = byte)	
PRINT@n, list	PRINT # 6, AT(X,Y);list	
PRINT # -1	CSAVE	Scrive dati su cassetta
RANDOM	RANDOMIZE	Cambia seme la funzione RND.

Appendice I
Conversione da Applesoft a ATARI BASIC II

GENERALITA'

L'Applesoft ha avuto origine dagli stessi sorgenti del BASIC dei calcolatori PET per cui esistono pochi problemi collegati al linguaggio vero e proprio nella conversione a Microsoft BASIC II ATARI.

DIFFERENZE

1 - La Apple ha aggiunto due funzioni di linguaggio ad Applesoft per migliorarne la compatibilità con l'integer BASIC della Apple stessa. Le funzioni sono ONERR, utilizzata per l'identificazione degli errori, e POP per cancellare valori dallo stack di ritorno dei sottoprogrammi (GOSUB).

 - ONERR può essere facilmente convertito in ON ERROR nel Microsoft BASIC II Atari.

 - POP non ha equivalente in quanto permette una forma di programmazione non strutturata nella quale le subroutines non sono delle subroutines vere e proprie. Per ottenere una conversione, è necessario aggiungere una variabile di servizio.
 A questo punto si sostituisce al POP una istruzione che ponga ad 1 la variabile, si emette un RETURN, allo statement successivo alla chiamata di subroutine (GOSUB) si pone un controllo che verifica il valore della variabile: se il valore è "Ø" nulla si altera nel flusso logico, se è "1" si salta alla istruzione desiderata dopo aver riazzerato la variabile.

2 - La dimensione del video Apple, assunta come standard è diversa da quella del video Atari.
 Menu e tabelle costruite per utilizzare completamente il video devono essere modificate.

3 - Le procedure di disco e funzioni di I/O delle periferiche Apple sono speciali.
 Stampe indirizzate a canali specifici vengono sfruttate per attivare partivcolari schede inseribili negli slot per le periferiche.
 Tutte le funzioni di stampa determinate dalle schede devono essere riprogrammate.

4 - La difficoltà più grande durante la conversione è rappresentata dalle modifiche necessarie agli statements relativi alle funzioni di grafica e del suono. Il dimensionamento totale del video ad alta risoluzione della Apple è di 280 colonne per 192 righe.
La tecnica utilizzata per il controllo del colore non è totalmente convenzionale, poichè ciascun pixel non può assumersi, in modo indipendente, il controllo di tutte le variabili del colore.
Il canale che comanda il suono è realizzato a singolo bit.

Appendice J
Conversione da BASIC 8K ATARI a ATARI Microsoft BASIC II

GENERALITA'

Le prestazione grafiche del Microsoft BASIC II ATARI sono più elevate rispetto a quelle del BASIC a 8K. Occorre perciò riscrivere la sezione dedicata alla grafica per poter usufruire a pieno delle capacità inserite nella funzione giocatore e missile. I registri SETCOLOR sono stati modificati in modo tale che i registri 0,1,2 e 3 abbiano ora un riferimento diretto a giocatore e missile. Quello che prima era SETCOLOR 0, cc e ll ora è diventato SECOLOR 4, cc e ll.

I numeri SETCOLOR sono stati combinati in modo tale che i numeri 0, 1,2,3 e 4 utilizzati per l'assegnazione dei registri sono ora 4,5,6,7 e 8. Altre modifiche comprendono l'istruzione FILL e l'istruzione PLOT concatenato, che sostituisce, a tutti gli effetti il DRAWTO.

Il Microsoft ha migliorato capacità di gestione delle stringhe. Se il programma originale occupa troppa memoria RAM, può essere notevolmente compattato riscrivendolo in Microsoft.

DIFFERENZE
Esistono piccole differenze da considerare, collegate all'istruzione RND() ed altre, prima di iniziare un lavoro di conversione in Microsoft BASIC II.
RND() può lavorare in modo identico alla RND del BASIC 8K se si inserisce uno statement RANDOMIZE come parte del programma.

Tutti i programmi che sono stati listati in BASIC 8K su dischetto, possono essere caricati con il Microsoft BASIC II Atari, con poche modifiche.

BASIC 8K ATARI	MICROSOFT BASIC ATARI	COMMENTI
ADR(VAR$)	VARPTR(VAR$) + 1	Identici
CLR	CLEAR	
DEG	-------	Non esiste equivalenza

BASIC 8K ATARI	MICROSOFT BASIC ATARI	COMMENTI
PLOT X,Y DRAWTO A,B	PLOT X,Y TO X,Y	Traccia una riga sullo schermo TV nei modi grafici. Lista un programma, usare un trattino per determinare l'intervallo dei numeri.
LIST mm,nn	LIST mm-nn	
LOCATE X,Y,VAR	VAR=ASC(SCRN$(X,Y))	Localizza il valore su registro
LPRINT "Hello"	OPEN # 7, "P:" OUTPUT PRINT # 7, "Hello"	Stampa "Hello" sulla stampante.
OPEN # iocb aexp1,aexp2 filespec	OPEN # iocb filespec INPUT	
POINT # iocb settore,byte	INPUT # iocb, AT (settore,byte) o PRINT # iocb at (S.E.)	
POP	-------	Usare la funzione USR per richiamare una routine in linguaggio macchina. Eseguire il POP in linguaggio macchina 6502. 6502.
POSITION X,Y PRINT # 6	PRINT # 6, AT(X,Y)	
SOUND voce frequenza, rumore, volume	SOUND voce frequenza, rumore, volume,durata	La durata è un'opzione nuova. E' fornita in sessantesimi di secondo. Perciò SOUND funziona nello stesso modo quando si convertono programmi in Microsoft BASIC II
TRAP exp	ON ERROR exp	Non esistono differenze
USR(addr,list)	USR(addr,pointer)	Il Microsoft BASIC II Atari accetta un solo argomento piuttosto che una lista di argomenti.
XIO 18	FILL X1,Y1 TO X2,Y2	L'istruzione FILL della Microsoft disegna una sequenza di punti dalle coordinate X1,

BASIC 8K ATARI	MICROSOFT BASIC ATARI	COMMENTI
		Y1 alle coordinate X2, Y2, eseguendo poi una scansione verso destra mentre riempie l'area delimitata da X1,Y1 e X2, Y2. La scansione verso destra si arresta ed inizia una nuova scansione di riempimento quando incontra una riga intera.

Il Microsoft BASIC II Atari esegue la funzioni di PADDLE, PTRING, STICK, STRIG tramite i comandi di PEEK e POKE. Vedere il Capitolo 6 - "Funzioni dei Giochi" per informazioni più dettagliate.

Note

Appendice K
Conversione Codici ATASCII ed Esadecimali in Decimali

CODICE DECIMALE	CODICE ESADECIMALE	CARATTERE	CODICE DECIMALE	CODICE ESADECIMALE	CARATTERE	CODICE DECIMALE	CODICE ESADECIMALE	CARATTERE	CODICE DECIMALE	CODICE ESADECIMALE	CARATTERE
0	0		15	F		30	1E	←	45	2D	-
1	1		16	10		31	1F	→	46	2E	.
2	2		17	11		32	20	SPAZIO	47	2F	/
3	3		18	12		33	21	!	48	30	0
4	4		19	13		34	22	"	49	31	1
5	5		20	14		35	23	#	50	32	2
6	6		21	15		36	24	$	51	33	3
7	7		22	16		37	25	%	52	34	4
8	8		23	17		38	26	&	53	35	5
9	9		24	18		39	27	'	54	36	6
10	A		25	19		40	28	(	55	37	7
11	B		26	1A		41	29	)	56	38	8
12	C		27	1B	ESC	42	2A	*	57	39	9
13	D		28	1C	↑	43	2B	+	58	3A	
14	E		29	1D	↓	44	2C	,	59	3B	;

CODICE DECIMALE	CODICE ESADECIMALE	CARATTERE	CODICE DECIMALE	CODICE ESADECIMALE	CARATTERE	CODICE DECIMALE	CODICE ESADECIMALE	CARATTERE	CODICE DECIMALE	CODICE ESADECOMALE	CARATTERE
60	3C	<	78	4E	N	96	60		114	72	r
61	3D	=	79	4F	O	97	61	a	115	73	s
62	3E	>	80	50	P	98	62	b	116	74	t
63	3F	?	81	51	Q	99	63	c	117	75	u
64	40	@	82	52	R	100	64	d	118	76	v
65	41	A	83	53	S	101	65	e	119	77	w
66	42	B	84	54	T	102	66	f	120	78	x
67	43	C	85	55	U	103	67	g	121	79	y
68	44	D	86	56	V	104	68	h	122	7A	z
69	45	E	87	57	W	105	69	i	123	7B	
70	46	F	88	58	X	106	6A	j	124	7C	|
71	47	G	89	59	Y	107	6B	k	125	7D	
72	48	H	90	5A	Z	108	6C	l	126	7E	
73	49	I	91	5B	[	109	6D	m	127	7F	
74	4A	J	92	5C	\	110	6E	n	128	80	
75	4B	K	93	5D	]	111	6F	o	129	81	
76	4C	L	94	5E	^	112	70	p	130	82	
77	4D	M	95	5F	_	113	71	q	131	83	

CODICE DECIMALE	CODICE ESADECIMALE	CARATTERE	CODICE DECIMALE	CODICE ESADECIMALE	CARATTERE	CODICE DECIMALE	CODICE ESADECIMALE	CARATTERE	CODICE DECIMALE	CODICE ESADECIMALE	CARATTERE
132	84		150	96		168	A8	(	186	BA	:
133	85		151	97		169	A9	)	187	BB	;
134	86		152	98		170	AA	*	188	BC	<
135	87		153	99		171	AB	+	189	BD	=
136	88		154	9A		172	AC	,	190	BE	>
137	89		155	9B		173	AD	-	191	BF	?
138	8A		156	9C		174	AE	.	192	C0	@
139	8B		157	9D		175	AF	/	193	C1	A
140	8C		158	9E		176	B0	0	194	C2	B
141	8D		159	9F		177	B1	1	195	C3	C
142	8E		160	A0		178	B2	2	196	C4	D
143	8F		161	A1	!	179	B3	3	197	C5	E
144	90		162	A2	"	180	B4	4	198	C6	F
145	91		163	A3	#	181	B5	5	199	C7	G
146	92		164	A4	$	182	B6	6	200	C8	H
147	93		165	A5	%	183	B7	7	201	C9	I
148	94		166	A6	&	184	B8	8	202	CA	J
149	95		167	A7	'	185	B9	9	203	CB	K

CODICE DECIMALE	CODICE ESADECIMALE	CARATTERE	CODICE DECIMALE	CODICE ESADECIMALE	CARATTERE	CODICE DECIMALE	CODICE ESADECIMALE	CARATTERE	CODICE DECIMALE	CODICE ESADECIMALE	CARATTERE
204	CC	L	222	DE	^	240	F0	p	248	F8	x
205	CD	M	223	DF	_	241	F1	q	249	F9	y
206	CE	N	224	E0	◆	242	F2	r	250	FA	z
207	CF	O	225	E1	a	243	F3	s	251	FB	♠
208	D0	P	226	E2	b	244	F4	t	252	FC	|
209	D1	Q	227	E3	c	245	F5	u	253	FD	↰
210	D2	R	228	E4	d	246	F6	v	254	FE	◀
211	D3	S	229	E5	e	247	F7	w	255	FF	▶
212	D4	T	230	E6	f						
213	D5	U	231	E7	g						
214	D6	V	232	E8	h						
215	D7	W	233	E9	i						
216	D8	X	234	EA	j						
217	D9	Y	235	EB	k						
218	DA	Z	236	EC	l						
219	DB	[	237	ED	m						
220	DC	\	238	EE	n						
221	DD	[	239	EF	o						

NOTE:

1) ATASCII indica ATARI ASCII. Le lettere e i numeri hanno gli stessi valori dei codici ATASCII, ma alcuni caratteri speciali sono differenti.

2) I caratteri da 128 a 255 sono riservati ai colori da 1 a 127.

3) Aggiungendo 32 al codice di un carattere maiuscolo si ottiene sempre il corrispondente carattere minuscolo.

4) Per ottenere i codici ATASCII occorre richiedere al computer (in modo diretto) la stampa ATASCII nel seguente modo: PRINT ASC ("______"). Inserire nello spazio la lettera, il carattere o il numero. Le virgolette sono obbligatorie.

Appendice L
Routines di Richiamo CIOUSR

Il Dischetto di Estensione del Microsoft BASIC II Atari contiene tre routines USR che forniscono un modo flessibile per interagire con le funzioni centrali di input/output (CIO) dell'Home Computer Atari. Queste routines, o altre simili scritte dall'Utente, permettono al programma BASIC di inviare o ricevere dati direttamente sotto la forma di un blocco di controllo input/output (IOCB).

Gli IOCB sono descritti dettagliatamente nel manuale "Atari Home Computer System - Note Tecniche di Riferimento". Sullo stesso documento si può trovare una descrizione completa della funzionalità di CIO.

Queste routines permettono al programmatore BASIC di eseguire attività come il reperimento di una directory disco, la formattazione di un dischetto, o la definizione di un determinato IOCB, e del numero di unità logica ad esso associato, per interfacciare con una unità RS-232.

Qui di seguito viene fornita una breve descrizione per l'utilizzo in programmi utente.

- PASSO 1. Inserimento delle routines in un Programma BASIC.

Tutte e tre le routines sono contenute nel file CIOUSR del dischetto di estensione del Microsoft BASIC II Atari. Esse hanno un formato leggibile dalla macchina e sono pronti, per essere trasferiti direttamente nella memoria RAM tramite POKE. Per assegnare la RAM per questo scopo, bisogna usare lo statement OPTION RESERVE n, dove n dovrebbe essere almeno 160. L'indirizzo di inizio dell'area riservata si ottiene con lo statement ADDR=VARPTR(RESERVE).

Inserire le routines nel programma BASIC seguendo le seguenti istruzioni:

```
10 OPEN # 1, "D:CIOUSR" INPUT
20 FOR T = 0 TO 159
30 GET #1,A
40 POKE ADDR+T,A NEXT I
50 CLOSE # 1
```

● PASSO 2. Creazione di matrici di dati

Le routines si trovano ora nell'area riservata del programma BASIC. Ci sono tre routines chiamate PUTIOCB, CALLCIO e GETIOCB.

PUTIOCB inizia alla posizione di indirizzo RAM definita ADDR.

CALLCIO inizia all'indirizzo ADDR + 61.

GETIOCB inizia all'indirizzo ADDR + 81.

La routines GETIOCB reperisce i bytes che possono essere modificati dall'utente da un IOCB specificato e li inserisce in una matrice intera di lunghezza 10. Questi parametri possono essere modificati ed i valori reinseriti nell'IOCB in un secondo tempo con la routine PUTIOCB. Una volta stabiliti i parametri adeguati, l'uso di CALLCIO fa sì che i valori IOCB siano eseguiti dalla funzione CIO.

Successivamente bisogna dimensionare una matrice intera per il reperimento e la memorizzazione di parametri IOCB. Tale matrice dovrebbe essere dimensionata a 10, usando l'opzione BASE fissato a zero. Segue un elenco degli elementi della matrice e dell'uso di ognuno di essi:

Numero elemento	Parametro IOCB
0	Questo elemento è il numero dell'IOCB da usare (da 1 a 8)
1	CODICE COMANDO
2	STATO - riportato
3	INDIRIZZO DEL BUFFER
4	LUNGHEZZA DEL BUFFER
5-10	Byte ausiliari da 1 a 6

Ogni elemento di una matrice intera ha due bytes per la memorizzazione dei dati, perciò l'indirizzo del buffer nell'elemento 3 è contenuto in un singolo elemento intero.

● PASSO 3. Richiamo delle routines USR

Un richiamo USR viene usato per eseguire le routines CIOUSR. La routine GETIOCB restituisce al programma i valori effettivi dei parametri IOCB specificati. Dopo aver modificato tali parametri nella matrice, si possono eseguire alcune funzioni di CIO (come, ad esempio, stabilire la velocità di trasferimento dei dati su un bocchettone RS-232) richiamando la routine PUTIOCB per inserire i valori desiderati nell'IOCB specificato. La routine CALLCIO viene successivamente richiamata per far eseguire la funzione CIO. Segue la sintassi necessaria per richiamare ognuna delle routines:

nvar = USR(indir,VARPTR(matrice(0)))

dove

nvar = Variabile numerica che riceve lo stato della funzione CIO nel caso di una chiamata della routine CALLCIO. Altrimenti non è specificatamente interessata da queste routines.

indir = Indirizzo iniziale dell'esatta routine CIOUSR (nell'esempio attuale tali indirizzi saranno ADDR per PUTIOCB, ADDR+16 per CALLCIO e ADDR+81 per GETIOCB).

matrice(0) = La matrice è una matrice intera usata dal programma per la ricerca e la memorizzazione di dati per le routines. Passando VARPTR di elemento 0 di questa matrice alle routines, rifornisce a queste routines l'indirizzo iniziale per la ricerca dei dati, partendo con il numero IOCB.

Il seguente programma rappresenta un esempio per l'utilizzo di una porta specializzata alle telecomunicazioni in aderenza allo Standard RS-232. Un altro esempio sull'uso di queste routines è inserito nella Appendice A (Disk Directory Program). Perchè il programma possa funzionare in modo corretto, l'unità che pilota L'RS-232 (nome file RS232.SYS su dischetto programma) deve essere caricata durante l'accensione dell'Home Computer Atari. Per caricare l'unità di pilotaggio, copiare il file RS232.SYS accodandolo al file BASIC AUTORUN.SYS:

CALCOLATORE: SELECT ITEM OR RETURN FOR MENU
(fare una selezione o tornare al menu (RETURN))

UTENTE: C RETURN

CALCOLATORE COPY..FROM,TO?
(copiare ...da, a?)

UTENTE: RS232.SYS,AUTORUN.SYS/A RETURN

CALCOLATORE: TYPE "Y" IF OK TO USE PROGRAM AREA
CAUTION: A "Y" INVALIDATES MEM.SAV
(batti "Y" se vuoi usare l'area programma
ATTENZIONE una "Y" rende non valido l'uso di MEM.SAV)

UTENTE: Y RETURN

```
100 !
110 !ROUTINE PER INIZIALIZZARE BOCCHETTONE RS-232
120 !
130 :                                    !STABILIRE GLI INDI
                                         RIZZI DELLE ROUTINES
                                         CIOUSR
140 DIM CIO%(10),S%(10)
```

```
145 S%(0)=5:S%(1)=&00
150 OPTION RESERVE 200
160 ADDR=VARPTR(RESERVE)          !SALVARE L'INDIRIZZO
                                  DEL BUFFER
170 PUTIOCB=ADDR                  !SALVARE L'INDIRIZZO
                                  DELLE ROUTINES
                                  CIOUSR
180 CALLCIO=ADDR+61
190 GETIOCB=ADDR+81
200 OPEN #1,"D:CIOUSR"INPUT       !TRASFERIRE LE ROUTI-
                                  NES CIOUSR NELLA RAM
210 FOR I=0 TO 159
220 GET#1,D:POKE ADDR+I,D
230 NEXT I
240 CLOSE#1
250 !                             !STABILIRE I  PARAME-
                                  TRI DI CONFIGURAZIO-
                                  NE
260 OPEN#1,"K:"INPUT
270 CIO%(0)= 2                    !NUMERO IOCB
280 CIO%(1)= 3                    !CODICE COMANDO  PER
                                  APERTURA
290 Y=VARPTR(CIO%(3))             !INDIRIZZO BUFFER
300 FSPEC$="R:"                   !NUMERO DI BOCCHETTO-
                                  NE
310 Z= VARPTR(FSPEC$)
315 Y=VARPTR(CIO%(3))
320 POKE Y,PEEK(Z+2)              !LOCAZIONE DI  AVVIO
                                  DEL BUFFER DELL'UNI-
                                  TA'
330 POKE Y+1,PEEK(Z+1)
335 Y=VARPTR(S%(3))
340 CIO%(5)=13                    !PARAMETRO AUXI
350 !
360 A=USR(PUTIOCB,VARPTR(CIO%(0)))   !STABILIRE IOCB
370 A=USR(CALLCIO,VARPTR(CIO%(0)))   !ESEGUIRE FUNZIONE
                                     CIO
380 A=USR(GETIOCB,VARPTR(CIO%(0)))
390 !
400 CIO%(1)=40                    !AVVIARE I/O   SIMUL-
                                  TANEO
410 CIO%(5)=13                    !PORRE I BYTES   AUX
                                  A ZERO
470 A=USR(PUTIOCB,VARPTR(CIO%(0)))   !MODIFICARE     IOCB
480 A=USR(CALLCIO,VARPTR(CIO%(0)))   !ESEGUIRE FUNZIONE
                                     CIO
      :
```

```
490 A=USR(PUTIOCB,VARPTR(S%(0)))
500 !                                  !AVVIARE LA TRASMIS-
                                       SIONE
520 PRINT"STARTING LOOP"
530 GET #1,A
540 PUT #2,A
550 POKE 764,255                       !AZZERARE BUFFER DEL-
                                       LA TASTIERA
560 X=USR(CALLCIO,VARPTR(S%(0)))       !CONTROLLARE STATO
                                       DI RS-232
570 IF PEEK(747)=0 THEN 600            !NESSUN CARATTERE
                                       INVIATO
580 GET#2,D                            !INVIARE CARATTERE
590 IF D<>10  THEN PRINT CHR$(D);
600 IF PEEK(764)<> 255 THEN 530        !CARATTERE INVIATO
                                       DALLA TASTIERA
610 GOTO 560
```

Per eseguire successive chiamate CIO si possono ripetere le righe 400-490 con diversi parametri IOCB nella matrice CIO%.

Appendice M
Eventi che si Verificano alla Fine di un Programma

AZIONI INTRAPRESE

Tasto premuto o statement eseguito	Chiude tutti i files	Azzera lo stack	Spegne il suono
STOP ERRORS BREAK	NO	NO	SI
Uscita dall'ultimo statement o "END"	SI	SI	SI
Dopo uno statement in modo diretto	NO	SI	NO
RUN	SI	NO	SI

N O T E

1. ATASCII significa "ATARI ASCII". Le lettere e i numeri hanno lo stesso valore di quelli in ASCII, ma alcuni caratteri speciali sono diversi.
2. Aggiungere 32 al codice delle lettere maiuscole per ottenere il codice delle maiuscole per la stessa lettera.
3. Per ottenere il codice ATASCII usare l'istruzione PRINT ASC (" ") in modo diretto. Riempire lo spazio vuoto con la lettera, il carattere o il numero di codice richiesto. Bisogna usare le virgolette!.
4. La visualizzazione normale viene fornita come la rappresentazione inversa del simbolo presente sul tasto della tastiera; cioè i simboli sono bianchi e lo sfondo è nero.
   La visualizzazione inversa invece, rappresenta i simboli in nero su di uno sfondo bianco.

Appendice N
Elenco Alfabetico di Parole Riservate

PAROLA RISERVATA	BREVE DESCRIZIONE DELLO STATEMENT BASICII
ABS	La funzione restituisce il valore assoluto (senza segno) della variabile o dell'espressione. Esempio: Y = ABS(A+B)
AFTER	Provoca l'immissione di un dato nello stack di INTERRUPT a tempo. Il tempo trascorso da associare con l'interruzione del tempo è indicato dall'e spressione numerica espressa in sessantesimi di secondo. Esempio: AFTER(180) GOTO 1000
AND	Operatore logico l'espressione è vera se ambedue le sottoespressioni collegate da AND sono vere. Esempio: IF A = 10 AND B = 30 THEN END
ASC	La funzione di stringa restituisce il valore ATASCII numerico di un singolo carattere di stringa. Esempio: PRINT ASC(A$)
AT	Posiziona l'output su disco o su schermo mediante lo statement PRINT.
ATN	La funzione restituisce l'arcotangente di un numero o espressione in radianti. Esempio: PRINT ATN(A)
AUTO	Genera automaticamente numeri di riga.
BASE	Usato con lo statement OPTION, stabilisce il valore minimo dell'indice di una matrice. Esempio: OPTION BASE1

PAROLA RISERVATA	BREVE DESCRIZIONE DELLO STATEMENT BASIC II
CHR	Usato con lo statement OPTION, assegna memoria RAM per insiemi alternativi di caratteri, dove - CHR1 riserva: 1024 bytes (128 caratteri), - CHR2 riserva: 512 bytes (64 caratteri), - CHR0 libera la memoria RAM assegnata Esempio : OPTION CHR1
CHR$	La funzione di stringa restituisce una stringa di un solo carattere equivalente ad un valore numerico compreso tra 0 e 255, in codice ATASCII. Esempio : PRINT CHR$(48)
CLEAR	Annulla tutte le stringhe ed azzera tutte le variabili. Esempio: CLEAR
CLEAR STACK	Azzera tutte le immissioni nello stack dei tempi. Esempio : CLEAR STACK
CLOAD	Carica programmi contenuti su cassetta nella memoria del calcolatore.
CLOSE	Lo statement I/O chiude un file dopo la conclusione delle operazioni di I/O. Esempio : CLOSE # 6
CLS	Cancella la parte testo dello schermo e posiziona il registro del colore di sfondo ad un valore specificato quando è presente. Esempio : CLS 35
COLOR	Stabilisce il registro del colore o il carattere che i successivi statement FILL e PLOT devono produrre. Esempio : COLOR 2
COMMON	E' uno statement di programma che passa variabili ad un programma concatenato con esso. Esempio : COMMON A,B,C$

PAROLA RISERVATA	BREVE DESCRIZIONE DELLO STATEMENT BASIC II
CONT	Riprende l'esecuzione del programma dopo un BREAK o uno STOP. Esempio : CONT
COS	La funzione restituisce il coseno di una variabile o di una espressione (gradi o radianti). Esempio : A = COS(2.3)
CSAVE	Salva su cassetta i programmi che sono contenuti nella memoria del calcolatore. Esempio : CSAVE
DATA	E' uno statement di I/O che elenca i dati da usare in uno statement READ. Esempio : DATA 2.3,"PLUS",4
DEF	Lo statement ha due applicazioni : 1 - Definisce una funzione aritmetica o di stringa Esempio : DEF SQUARE(X,Y)=SQR(X*X+Y*Y) 2 - Definisce la variabile standard di tipo INT, SNG, DBL o STR. Esempio : DEFINT I-N
DEL	Cancella righe di programma Esempio : DEL 20-25
DIM	Riserva un determinato spazio di memoria per matrici, vettori o vettori di stringa. Esempio : DIM A(3), B$(10,2,3)
END	Arresta il programma, chiude tutti i files e ritorna al livello di comando BASIC. Esempio : END
EOF	Restituisce il valore "vero" (-1) se il file è posizionato alla fine. Esempio : IF EOF(1) GOTO 300

PAROLA RISERVATA	BREVE DESCRIZIONE DELLO STATEMENT BASIC II
ERL	Restituisce il numero della riga con errore Esempio : PRINT ERL
ERR	Restituisce il codice di errore. Esempio : IF ERR=62 THEN END
ERROR	Genera il codice di errore. (Vedi Appendice O). Può richiamare la routine ON ERROR dell'utente o costringe il BASIC a gestire l'errore. Esempio : ERROR 17
EXP	Eleva la costante e alla potenza dell'espressione. Esempio : B = EXP(3)
FILL	Riempie di un colore l'area compresa tra due punti. Esempio : FILL 10,10 TO 20,20
FOR...TO...STEP	Usato con lo statement NEXT ripete una sequenza di righe di programma. La variabile è incrementata del valore di STEP. Esempio : FOR DAY=1 TO 5 STEP 2
FRE(0)	Fornisce lo spazio libero in memoria al programmatore. Esempio : PRINT FRE(0)
GET	Legge un byte da un'unità di input. Esempio : GET # 1,D
GOSUB	Salta ad una subroutine che inizia al numero di riga specificato. Esempio : GOSUB 210
GOTO	Salta ad un numero di riga specificato. Esempio : GOTO 90
GRAPHICS	Stabilisce quali liste di visualizzazione (display lists) e modi grafici contenuti nel sistema operativo devono essere usati per produrre una visualizzazione a video. Esempio : GRAPHICS 5

PAROLA RISERVATA	BREVE DESCRIZIONE DELLO STATEMENT BASIC II
IF...THEN	Se l'espressione è vera, viene eseguita là clausola THEN. Altrimenti viene eseguito lo statement successivo. Esempio : IF ENDVAL > 0 THEN GOTO 200.
IF...THEN...ELSE	Se l'espressione è vera, viene eseguita la clausola THEN. Altrimenti viene eseguita la clausola ELSE o lo statement successivo. Esempio : IF X< Y THEN Y=X ELSE Y=A
INKEY$	Restituisce una stringa di un solo carattere letta dalla tastiera o una stringa vuota se sulla tastiera non esiste alcun carattere in attesa. Esempio : A$=INKEY$
INPUT#	Legge dati da un'unità. Esempio : INPUT #1,A,B
INPUT	Legge dati dalla tastiera. Un segno di punto e virgola dopo INPUT sopprime la trasmissione al video del RETURN (ritorno carrello + nuova linea). Se viene fatta una richiesta commentata da una frase, questa appare a video; altrimenti appare un punto interrogativo. Esempio : INPUT "VALORI";A,B
INSTR	Restituisce la posizione numerica della prima apparizione di Y$ in X$ eseguendo la scansione, ad esempio, dal terzo carattere in X$. Esempio : INSTR(3,X$,Y$)
INT	Valuta l'espressione per determinare il più alto numero intero minore dell'espressione stessa. Esempio : C=INT(X+3)
KILL	Cancella un file dal disco. Esempio : KILL "D:INVEN.BAS"
LEFT$	Restituisce il numero di caratteri indicato a partire da sinistra nell'espressione di stringa.

PAROLA RISERVATA	BREVE DESCRIZIONE DELLO STATEMENT BASIC II
	Esempio : B$=LEFT$(X$,8)
LEN	Restituisce la lunghezza della stringa indicata, in bytes o caratteri (1 byte contiene 1 carattere). Esempio : PRINT LEN(B&)
LET	Assegna un valore ad una variabile specificata. Esempio : LET X=I+5
LINE INPUT	Legge un'intera riga della tastiera. Il punto e virgola dopo LINE INPUT sopprime la trasmissione del RETURN. Vedere INPUT. Esempio : LINE INPUT "NAME";N$
LIST	Visualizza, stampa o memorizza il listing del programma. Esempio : LIST 100-1000
LOAD	Carica in memoria un file-programma. Esempio : LOAD "D:INVEN"
LOCK	Blocca il file indicato nell'espressione di stringa Esempio : LOCK "D1:TEST.BAS"
LOG	Restituisce il logaritmo naturale di un numero. Esempio : D=LOG(Y-2)
MERGE	Esegue la fusione tra un programma su disco e il programma in memoria, per numero di riga. Esempio : MERGE "D:SUB1"
MID$	Estrae caratteri dal mezzo di una stringa iniziando dalla posizione indicata fino alla fine della stringa o per il numero specificato di caratteri. Esempio : A$=MID$(X5,5,10)
MOVE	Sposta bytes da un'area all'altra di memoria, senza cambiare il blocco trasferito. Esempio : MOVE 45000.50000.6

PAROLA RISERVATA	BREVE DESCRIZIONE DELLO STATEMENT BASIC II
NAME	Cambia il nome di un file su disco. Esempio : NAME"D:OLDFILE" TO "NEWFILE"
NEW	Cancella il programma e le variabili correnti. Esempio : NEW
NEXT	Fa si che un loop FOR/NEXT termini o continui a seconda del valore di particolari variabili o espressioni. Esempio : NEXT I
NOT	Operatore unario, usato nelle espressioni logiche. Indica un valore 0 se l'espressione è diversa da 0 e valore 1 se l'espressione è 0. Esempio : IF A=NOT B
NOTE	Fa sì che l'attuale numero di settore di disco sia memorizzato nella prima variabile e il numero di byte nella seconda variabile, per il file associato con l'IOCB . Esempio : NOTE #1,S,B
ON ERROR	Abilita la subroutine di ricerca dell'errore, iniziando dalla riga specificata. Se la riga = 0 disabilita la ricerca dell'errore. Se riga = 0 all'interno della subroutine di ricerca dell'errore, costringe l BASIC a gestire l'errore. Esempio : ON ERROR GOTO 1000
ON... GOSUB	Esegue un salto a subroutine allo statement indicato dal valore dell'espressione (se l'espressione = 1 salta allo statement 20; se l'espressione = 2 salta a 20; se l'espressione = 3 salta a 40; altrimenti emette l'errore. Esempio : ON DATE% +1 GOSUB 20,20,40
ON...GOTO	Salta allo statement indicato dall'espressione (se INDEX = 1 salta a 20; se INDEX = 2 a 30; se INDEX 3 a 40; altrimenti emette errore. Esempio : ON INDEX GOTO 20,30,40

PAROLA RISERVATA	BREVE DESCRIZIONE DELLO STATEMENT BASIC II
OPEN	Apre un'unità. Il modo deve essere uno dei seguenti: INPUT, OUTPUT, UPDATE e APPEND. Esempio : OPEN #1,"D:INVEN.DAT" OUTPUT
OPTION BASE	Dichiara il valore iniziale per gli indici delle matrici; "n" e 0 o 1. Esempio : OPTION BASE 1
OPTION CHR	Assegna spazio per insiemi di caratteri alternativi. Esempio : OPTION CHR1
OPTION PLM	Assegna spazio per i grafici giocatore e missile. Esempio : OPTION PLM1
OPTION RESERVE	Riserva spazio libero da usare in un programma in linguaggio assemblatore. Esempio : OPTION RESERVE (50)
OR	Operatore logico usato tra due espressioni. Se una delle due è vera, si ha come risultato un "1". Si ha invece "0" se ambedue le espressioni sono false. Esempio : IF A=10 OR B=30 THEN END
PEEK	La funzione restituisce la forma decimale del contenuto di una determinata posizione di memoria. Esempio : PRINT PEEK (&2000)
PLM	Usato con lo statement OPTION, assegna memoria RAM per la grafica giocatore e missile, dove: - PLM1 = risoluzione a riga singola - PLM2 = risoluzione a doppia riga - PLM0 = libera la RAM assegnata Esempio : OPTION PLM2
PLOT	Disegna un singolo punto sullo schermo o raccorda con una riga più punti. Esempio : PLOT10, 10 TO 20,20
POKE	Inserisce il byte specificato nella locazione di memoria indicata. Esempio : POKE &2310,255

PAROLA RISERVATA	BREVE DESCRIZIONE DELLO STATEMENT BASIC II
PRINT	Comando di I/O che causa l'output dal calcolatore sull'unità indicata. Esempio : PRINT USING"!";A$,B$
PUT	Scrive dati su un file Esempio : PUT#3,4
RANDOMIZE	Cambia seme al generatore di numeri casuali. Esempio : RANDOMIZE
READ	Legge i dati successivi in un elenco associato alla istruzione DATA e li assegna alle variabili indicate. Esempio : READ I,X,A$
REM	Nota: Permette di inserire commenti nel programma senza che il calcolatore li esegua. Forme alternative sono il punto esclamativo (!) e l'apostrofo('). Esempio : REM SOTTOPROGRAMMA UNO
RENUM	Rinumera le righe del programma. Esempio : RENUM 100,,100
RESERVE	Usato con lo statement OPTION, riserva un determinato numero di bytes per l'utente. Esempio : OPTION RESERVE (512)
RESTORE	Riposiziona il puntatore associato all'istruzione DATA per poter leggere i dati più di una volta. Esempio : RESTORE
RESUME	Provoca il ritorno del programma da ON ERROR, o dalla routine di INTERRUPT temporale, allo statement che ha provocato l'errore. RESUME NEXT ritorna allo statement successivo a quello che ha provocato l'errore e RESUME numero-riga ritorna allo statement specificato da numero-riga. Esempio : RESUME

PAROLA RISERVATA	BREVE DESCRIZIONE DELLO STATEMENT BASIC II
RETURN	Provoca il ritorno del programma dalla subroutine allo statement immediatamente dopo quello in cui era apparso GOSUB. Esempio : RETURN
RIGHT$	Estrae il numero specificato di caratteri da una stringa, a partire da destra. Esempio : C$ = RIGHT$(X$,8)
RND	Genera un numero casuale. Se il parametro è eguale 0 fornisce un numero casuale tra 0 e 1. Se il parametro è maggiore di 0, restituisce un numero casuale compreso tra lo 0 e il parametro. Esempio : E = RND(10)
RUN	Esegue un programma iniziando dal primo numero di riga. Esempio : RUN
SAVE	Salva il programma in memoria con il nome "filenome". SAVE "filenome" LOCK scrive il programma su disco e lo protegge da tentativi di lettura o modifica. Esempio : SAVE"D:PROG"
SCRN$	Il carattere o il numero di colore del "pixel" alle coordinate X e Y viene restituito come il valore di questa funzione, usando la funzione ASC. Esempio : A$ = ASC(SCRN$ (23,5))
SETCOLOR	Associa un colore e una luminosità ad un registro colore. Esempio : SETCOLOR 0.5.5
SGN	Statement che esprime il segno : 1 se l'espressione è più grande di 0 0 se l'espressione è uguale a 0 -1 se l'espressione è più piccola di zero Esempio : B = SGN(X + Y)

PAROLA RISERVATA	BREVE DESCRIZIONE DELLO STATEMENT BASIC II
SIN	La funzione restituisce il seno trigonometrico di un determinato valore espresso in gradi. Esempio : B = SIN(A)
SOUND	Statement che inizializza uno dei generatori del suono. Esempio : SOUND 1,121,8,10,60
SPC	Usato negli statement PRINT stampa degli spazi. Esempio : PRINT SPC(5),A$
SQR	La funzione restituisce la radice quadrata del valore indicato. Esempio : C=SQR(D)
STACK	Restituisce il numero di entrate disponibili nello stack dei tempi. Esempio : A = STACK
STATUS	La funzione accetta un singolo argomento numerico o di stringa e restituisce lo stato del file o del numero di unità logica. Esempio : ST = STATUS(2)
STOP	Arresta l'esecuzione,ma non chiude i files. Esempio : STOP
STR$	La funzione restituisce una stringa di caratteri uguale al valore numerico fornito. Esempio : PRINT STR$ (35)
STRING$	Restituisce una stringa composta da un determinato numero di duplicazioni di A$. Esempio : X$ = STRING$(100,"A") Restituisce una stringa lunga 100 unità contenente CHR$(65). Esempio : Y$ = STRING$(100,65)
TAB	Usato negli statements PRINT, esegue una tabulazione del carrello su determinate posizioni. Esempio : PRINT TAB(20),A$

PAROLA RISERVATA	BREVE DESCRIZIONE DELLO STATEMENT BASIC II
TAN	Restituisce la tangente dell'espressione in radianti. Esempio : D = TAN(3.14)
TIME	Restituisce la rappresentazione numerica del tempo derivata dall'orologio in tempo reale. Esempio : ATM = TIME
TIME$	Restituisce l'ora del giorno in una notazione di 24 ore nella stringa. Il formato è : HH:MM:SS: Esempio : TIME$ = "08:55:05" PRINT TIME$
TROFF	Disabilita la funzione di tracciatura. Esempio : TROFF
TRON	Abilita la funzione di tracciatura. Esempio : TRON
UNLOCK	Questo statement disattiva la condizione "LOCK" (di bloccaggio). Esempio : UNLOCK"D1:DATA.OUT"
USING	Fornisce il formato di stringa per l'output stampato. Esempio : PRINT USING"###.##; PDOLLARS
USR	La funzione riporta i risulatati di una subroutine in linguaggio macchina. Esempio : X = USR(SVBV, VARPTR(ARR(0)))
VAL	La funzione riporta il valore numerico equivalente di una stringa. Esempio : PRINT VAL("3.1")
VARPTR	Restituisce l'indirizzo di memoria di una variabile o di un grafico, o uno zero se alla variabile non è stato assegnato alcun valore. Esempio : I = VARPTR(X)

PAROLA RISERVATA	BREVE DESCRIZIONE DELLO STATEMENT BASIC II
VERIFY	Confronta il programma in memoria con quello sul file specificato. Se i due programmi non sono identitici restituisce un errore. Esempio : VERIFY "D1:DATA.OUT"
WAIT	Confronta l'uguaglianza e mette in attesa l'esecuzione finchè il risultato non sia uguale al terzo parametro. Esempio : WAIT &E456,&FF,30
XOR	Esegue l'OR esclusivo (intero) a livello di bit. Esempio : IF A XOR B = 0 THEN END

Appendice O
Codici di Errore

CODICE	COMMENTI A SCHERMO	ERRORE
1	NEXT WITHOUT FOR ERROR IN Line #	NEXT è stato usato senza il corrispondente statement FOR. Questo errore può verificarsi anche se gli statements NEXT sono in posizioni errate in un loop nidificato.
2	SYNTAX ERROR IN Line #	Errore di sintassi. Punteggiatura errata,parentesi non aperta/chiusa, caratteri non validi o parole chiave scritte male provocano errori di sintassi.
3	RETURN WITHOUT GOSUB ERROR	Uno statement RETURN è stato inserito prima del corrispondente COSUB.
4	OUT OF DATA ERROR IN Line #	Ad uno statement READ o INPUT non sono stati assegnati abbastanza dati. Può essere stato dimenticato uno statement DATA o tutti i dati sono stati letti da un'unità (dischetto o cassetta).
5	FUNCTION CALL ERROR IN Line #	Il programma ha tentato di eseguire un'operazione usando un parametro non valido. Esempio : radice quadrata di un numero negativo, o logaritmo di un numero negativo.
6	OVERFLOW	A seguito di un'operazione matematica o da input da tastiera, è risultato un numero troppo grande o troppo piccolo.
7	OUT OF MEMORY ERROR	Tutta la memoria disponibile è stata usata o riservata. Ciò può verificarsi con matrici aventi dimensioni molto grandi, salti nidificati come GOTO, GOSUB e loops FOR/NEXT.

CODICE	COMMENTI A SCHERMO	ERRORE
8	UNDEF'D LINE ERROR IN Line #	E' stato fatto un tentativo di far riferimento o saltare ad una riga non esistente ("non definita").
9	SUBSCRIPT ERROR IN Line #	E' stato assegnato un elemento di matrice al difuori della dimensione della matrice stessa.
10	REDEF'N ERROR IN Line #	Tentativo di dimensionare una matrice che è già stata dimensionata usando lo statement DIM o gli standard.
11	DIVISION BY ZERO	Non è valido usare zero nel denominatore.
12	ILLEGAL DIRECT ERROR	L'uso di INPUT, GET o DEF in modo diretto non è permesso.
13	TYPE MISMATCH ERROR IN Line #	Non è permesso assegnare una variabile di stringa ad una variabile numerica e viceversa.
14	FILE I/O ERROR	Errore generale di I/O.
15	QUANTITY TOO BIG ERROR IN Line #	La variabile di stringa è più lunga di 255 caratteri.
16	FORMULA TOO COMPLEX ERROR	Un operazione matematica o di stringa è risultata troppo complessa. Bisogna suddividerla in passi più brevi.
17	CAN'T CONTINUE ERROR	Un comando CONT, nel modo diretto, non può essere eseguito perchè il programma ha incontrato uno statement END.
18	UNDEF'D FUNCTION ERROR	La funzione USR non può essere eseguita. Il codice utente contiene un errore nella logica o il puntatore di inizio dell'USR indica un indirizzo di memoria errato.

CODICE	COMMENTI A SCHERMO	ERRORE
19	NO RESUME ERROR IN Line #	E' stata raggiunta la fine del programma durante la ricerca di un errore.
20	RESUME WITHOUT ERROR IN Line #	Si è incontrato RESUME prima dello statement ON ERROR GOTO.
21	FOR WITHOUT NEXT ERROR	E' stato incontrato lo statement NEXT prima di uno statement FOR.
22	LOCK ERROR	Tentativo di modificare o listare un programma salvato con l'opzione LOCK.
23	TIME ERROR	Interruzioni di tempo sono in conflitto tra di loro. Statements AFTER non seguiti da RESUME.

Per una spiegazione dei seguenti codici di errore, consultare il Manuale "ATARI Sistema Operativo a Dischi II - Manuale di Riferimento".

128	BREAK abort	Interruzione per mezzo di BREAK
129	IOCB	
130	Nonexisten device	Unità non esistente
131	IOCB write only	IOCB a sola scrittura
132	Invalid command	Comando non permesso
133	Device or file not open	Unità o file non aperto
134	Bad IOCB number	Numero di IOCB errato
135	IOCB read-only error	Errore di IOCB a sola lettura
136	EOF	Fine del file
137	Truncated record	Record troncato
138	Device timeout	Fine del tempo disponibile per la unità
139	Device NAK	NAK (non disponibilità) dell'unità
140	Serial bus	Bus seriale
141	Cursor out of range	Cursore fuori dai limiti
142	Serial bus data frame overrun error	Errore di sovrapposizione del pacchetto dati su bus seriale
143	Serial bus data frame checksum error	Errore nel totale di controllo del pacchetto dati su bus seriale
144	Device-done error	Errore dovuto all'unità
145	Read after write-compare error	Errore di lettura dopo confronto con scrittura

146	Function not implemented	Funzione non implementata
147	Insufficient RAM	RAM insufficiente
160	Drive number error	Numero unità errato
161	Too many OPEN files	Troppi files aperti
162	Disk full	Disco pieno
163	Unrecoverable system date I/O error	Errore non ripristinabile di I/O dei dati di sistema.
164	File number mismatch	Manca corrispondenza del numero del file
166	POINT data length error	Errore di lunghezza dei dati POINT
167	File locked	File bloccato
168	Command invalid	Comando non permesso
169	Directory full	Directory piena
170	File not found	File non trovato
171	POINT invalid	POINT non permesso

ATARI®


A Warner Communications Company

ATARI International (Italy) Inc.
Viale della Liberazione, 18
20124 MILANO